

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE ENGENHARIA GEOGRÁFICA, GEOFÍSICA E DA ENERGIA



Sea-Stars Watching: Uma aplicação móvel colaborativa para georreferenciação de estrelas-do-mar

João Carlos Guerra Gradiz

Mestrado em Sistemas de Informação Geográfica - Tecnologias e Aplicações

Dissertação orientada por:
Prof. Doutor João Catalão Fernandes e Prof. Doutora Ana Paula Afonso

2017

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE ENGENHARIA GEOGRÁFICA, GEOFÍSICA E DA ENERGIA



Sea-Stars Watching: Uma aplicação móvel colaborativa para georreferenciação de estrelas-do-mar

João Carlos Guerra Gradiz

Mestrado em Sistemas de Informação Geográfica - Tecnologias e Aplicações

Dissertação orientada por:
Prof. Doutor João Catalão Fernandes e Prof. Doutora Ana Paula Afonso

2017

Agradecimentos

Este projecto representa a conclusão de uma fase importante da minha formação académica ditando o fim de meu percurso na Faculdade de Ciências da Universidade de Lisboa. Aproveito para deixar os meus profundos agradecimentos às pessoas que mais me apoiaram e estiveram sempre presentes nesta fase a nível anímico e profissional e cuja presença contribuiu de forma importante e facilitou de alguma forma este longo caminho.

Em primeiro lugar quero agradecer aos meus orientadores Prof. Doutor João Catalão Fernandes e Prof. Doutora Ana Paula Afonso que me orientaram neste percurso. Quero agradecer ao Centro Português de Actividades Subaquáticas pela ideia e pela oportunidade de me concederem o desenvolvimento deste projecto. Aos meus colegas e amigos de empresa por todo o apoio e companheirismo no dia-a-dia, Luís Gonçalves, João Carmo, Joana Verde, Sara Flor, Sara Ferreira, Sisto Chaves, Telmo Silva, Marcos Lourenço, Rui Reis, Pedro Vicente, Alcino Loureiro, Xavier Gonçalves, Eduardo David, Jorge Pinto, Jorge Carvalho, Cristiana Oliveira, João Perneco e Bruno Figueiredo.

Agradeço aos meus colegas e amigos que fiz na faculdade. Foi muito importante o vosso contributo para o meu percurso académico. Obrigado André Adro, João Ferreira, Ion Donica, Cíntia Teles, André Teles, Romeu Silva, Miguel Santos, Pedro Teixeira, Nuno Paixão, Cláudia Aguiar e Mário Pinto. Agradeço aos meus amigos sempre presentes de Beja, Cuba e Lisboa, Vasco Tareco, João Negra, Rui Casaca e Francisco Caipirra. Agradeço também aos meus amigos de longa data e também sempre presentes Pedro Fontes, Rui Batista, Tiago Vargas.

Agradeço ao atletismo e aos meus amigos e colegas de treino enquanto treinei a modalidade, Hugo Sioga, David Palma, Nuno Alpiarça, Gabriel Potra, Luís Gonçalves, António Carvalho, Renato Moura, António Jonas, César Pedreira, Victor Zabumba, Ricardo Lemos, Edi Sousa, Tiago Silvestre, Ricardo Lima, André Lima e Carlos Fradão. Foi o atletismo e vocês que me fizeram crescer, ter disciplina e me ajudaram a tornar na pessoa que sou hoje.

Por último, quero agradecer à minha família, mãe, pai, sem vocês não teria conseguido chegar até aqui. Obrigado por todo o apoio, carinho e auxílio que me deram ao longo de todos estes anos. À minha irmã Sara está longe mas sempre por perto; Ao meu irmão António e aos meus sobrinhos que fazem de mim um tio babado; Aos meus primos Tiago, Susana, Paula, Maria João, Miguel e Cláudio; Ao meu tio Jorge, tia Micas, tia Rosa e tia Deolinda. À minha avó Lourdes Quintais, que graças a todos foi possível dar o meu salto para a minha vida em Lisboa proporcionando uma boa qualidade de vida.

Resumo

Esta dissertação descreve o projecto realizado por mim para o Centro Português de Actividades Subaquáticas, enquadrado no âmbito do projecto *Sea-Stars Watching* de Setembro de 2016 a Setembro de 2017 e tem como objectivo desenvolver uma aplicação *web* responsiva adaptada a dispositivos móveis como *tablets* e *smartphones*.

O projecto *Sea-Stars Watching* primeiramente designado como “Estrelas-do-mar em Águas Portuguesas”, visa averiguar o estado de conservação das espécies de estrelas-do-mar observando a presença ou ausência de sintomas degenerativos, surgindo a necessidade de criar uma aplicação que permitisse catalogar registos fotográficos.

Através desta aplicação é possível, além de averiguar o estado de conservação das espécies de estrelas-do-mar, analisar o estado de conservação de outras espécies animais e vegetais. Graças à capacidade de efectuar registos fotográficos e fazer a georreferenciação destes eventos, é criado um catálogo com os dados adquiridos. A área envolvida neste projecto é Portugal Continental e arquipélagos da Madeira e Açores.

De acordo com os requisitos, a aplicação acede a localização através do dispositivo de cada utilizador. Este, posteriormente tem acesso a um conjunto de funcionalidades onde lhe é permitido, criar uma conta, alteração de perfil pessoal, álbuns e fotografias associadas e, ter o acesso a uma base de dados geográfica sobre o conteúdo fotográfico georreferenciado dos utilizadores através de um mapa interactivo. A solução passou por desenvolver a aplicação através de uma *framework* específica para desenvolvimento *web*.

A aplicação foi testada através de testes e por um perito responsável pelo projecto. Estes testes foram essenciais para a validação do projecto, comprovando o bom funcionamento e utilidade.

Palavras-chave: georreferenciação, estrelas-do-mar, aplicação *web* responsiva, desenvolvimento *web*, framework

Abstract

This thesis describes the project carried out by me to the Portuguese Center for Underwater Activities, within the scope of the Sea-Stars Watching project from September 2016 to September 2017 and aims to develop a responsive web application adapted to mobile devices such as tablets and smartphones.

The project Sea-Stars Watching first designated as "Estrelas-do-mar em Águas Portuguesas", has the objective to find out the state of conservation of starfish species by observing the presence or absence of degenerative symptoms. For this purpose, an web application to catalog photographic records was needed.

Through this application it is possible, not only to analyze the state of conservation of the starfish species, but to analyze the state of conservation of other animal and plant species too. Thanks to the ability to make photographic records and georeferentiate these events, a catalog with the acquired data can be created. The area involved in this project is Portugal and the archipelagos of Madeira and the Azores.

According to the requirements, the application accesses the location through the device drivers that each user owns. Later this user has access to a set of features where he is allowed to create an account, change personal profile, creation of albums and associated photographs and have access to a geographic database on the georeferenced photographic content of users through a dynamic interactive map. The solution was to develop the application through a specific framework for web development.

The application was tested through unit tests and supervised by an expert in charge of the project. These tests were essential for the validation of the project, ascertaining that it's proper of use.

Keywords: Georeferencing, starfish, responsive web application, MVC, Laravel, data model, MySQL

Conteúdo

Lista de Figuras	x
Lista de Tabelas.....	xiii
Lista de Acrónimos	xv
Capítulo 1- Introdução	1
1.1. Motivação.....	1
1.2. Objectivo	1
1.3. Estado de Arte	2
1.4. A Instituição	2
1.5. Planeamento	3
1.6. Notação Adoptada	4
1.7. Estrutura da Dissertação.....	4
Capítulo 2 – Contexto do Projecto	5
2.1. Introdução	5
2.2. Solução Adoptada	5
2.2.1. SGBD	5
2.2.2. Servidor Web	6
2.2.3. Equipamento, Tecnologias e Ferramentas.....	6
2.3. Soluções Alternativas	8
2.3.1. Alternativa ao SGBD	8
2.3.2. Alternativa ao Servidor Web.....	9
2.3.3. Alternativas de Tecnologias	9
2.4. Sumário	10
Capítulo 3 – Trabalho realizado	11
3.1. Metodologia e Equipa	11
3.2. Arquitectura Geral.....	12

3.2.1.	Front-End e Back-End.....	12
3.2.2.	Esquema de Dados	13
3.2.3.	Componente Lógica de Desenvolvimento	15
3.2.4.	Estrutura Rest	18
3.2.5.	Middleware	23
3.3.	Ambiente de Desenvolvimento	25
3.3.1.	Instalação do Ambiente de Desenvolvimento	25
3.3.2.	Configuração da Máquina Virtual.....	27
3.3.3.	Comandos Específicos do Laravel	28
3.3.4.	Ligação à Base de Dados	29
3.3.5.	Migração do Modelo de Dados	31
3.4.	Funcionalidades da Aplicação Sea-Stars Watching	32
3.5.	Utilização dos Mapas Google	38
3.6.	Sumário	38
	Capítulo 4 – Testes e Avaliação por Perito	39
	Capítulo 5 - Conclusão.....	41
5.1.	Principais Contribuições	41
5.2.	Competências Adquiridas	41
5.3.	Dificuldades Encontradas.....	42
5.4.	Trabalho Futuro.....	42
	Bibliografia	43

Lista de Figuras

Figura 1 - Servidor <i>Web</i>	6
Figura 2 - Metodologia abordada no desenvolvimento do projecto.....	11
Figura 3 - Arquitectura geral da aplicação.	12
Figura 4 - Esquema UML do Modelo de dados.	14
Figura 5 - Padrão de arquitectura MVC.	15
Figura 7- Resposta HTTP em JSON com os dados de um utilizador.	23
Figura 8 - Página devolvida com a informação do Utilizador.....	23
Figura 9 – Camadas do Sistema Middleware.	24
Figura 10 - Todas as camadas do Sistema Middleware.	24
Figura 11 - Aplicação do <i>Middleware</i> no sistema de rotas.	25
Figura 12 - Instalação da aplicação via Composer.....	26
Figura 13 - Pastas e ficheiros da aplicação.....	26
Figura 14 - Configuração da máquina virtual.	27
Figura 15 - Mapeamento da máquina virtual.	28
Figura 16 - Configuração do driver da BD na aplicação.....	30
Figura 17 - Ficheiro <i>.env</i> de configuração.	30
Figura 18 - Execução do comando para migrar o modelo de dados.	31
Figura 19 - Resultados depois da migração do modelo de dados.....	31
Figura 20 - Criação de conta de utilizador.	32
Figura 21 - Pagina inicial.	32
Figura 22 - Criação de um álbum novo.....	33
Figura 23 - Descarregamento e visualização de fotografias.....	33
Figura 24 - Georreferenciação e edição da descrição da fotografia.	34

Figura 25 - Interacção com o mapa interactivo.....	34
Figura 26 - Álbum e perfil de outro utilizador.	35
Figura 27 - Envio e aceitação de pedidos de amizade.....	35
Figura 28 - Exemplo de alteração no perfil do utilizador.....	36
Figura 29 - Pesquisa e <i>logout</i>	36
Figura 30 - Vista de um ecrã com tamanho clássico.	37
Figura 31- Vista de um ecrã de smartphone.....	37

Lista de Tabelas

Tabela 1 - Planeamento do projecto.....	4
Tabela 2 - Tecnologias utilizadas.....	7
Tabela 3 - Controladores da aplicação.....	17
Tabela 4 - Métodos e operações CRUD.....	18
Tabela 5 - Estrutura REST de recursos gerais.....	19
Tabela 6 - Estrutura REST para autenticação e registo de utilizadores.....	19
Tabela 7 - Estrutura REST para renovação de <i>password</i>	19
Tabela 8 - Estrutura REST para alteração e visualização de perfis.....	20
Tabela 9 - Estrutura REST para gestão de albums de utilizadores.....	20
Tabela 10 - Estrutura REST para gestão de fotografias.....	21
Tabela 11 - Estrutura REST para pesquisa e interligação de utilizadores.....	22
Tabela 12 - Estrutura REST para recolha das fotografias georreferenciadas.....	22
Tabela 13 - Lista de comandos mais utilizados.....	29

Lista de Acrónimos

CPAS - Centro Português de Actividades Subaquáticas

SGBD - Sistema de Gestão de Base de Dados

SQL - *Structured Query Language*

HTTP - *Hiper-Text Transfer Protocol*

HTML - *Hyper-Text Markup Language*

PHP - *Hypertext Preprocessor*

BD - Base de Dados

JSON - *JavaScript Object Notation*

CSS - *Cascading Style Sheet*

VB - *Visual Basic*

IIS - *Internet Information Services*

API – *Application Programming Interface*

REST - *REpresentational State Transfer*

JSP – *Java Server Page*

ASP – *Active Server Page*

CRUD – *Create Read Update and Delete*

IP – *Internet Protocol*

MVC – *Model-View-Controller*

Capítulo 1- Introdução

Neste primeiro capítulo é apresentado o contexto do projecto começando pela abordagem motivacional, seguida do objectivo, a caracterização da instituição para a qual foi desenvolvido este trabalho, o planeamento do projecto e a estrutura da dissertação.

1.1. Motivação

Actualmente a equipa do Centro Português de Actividades Subaquáticas (CPAS) apenas consegue recolher informação de algumas espécies de estrelas-do-mar que após observação no local, recolha de informação, fotografias, registo de data de observação, é enviado para o CPAS. Contudo, actualmente esta situação cria alguns problemas nos dados recolhidos devido a escassez ou falta de precisão dos mesmos.

Com o crescimento das aplicações móveis participativas, nas quais os cidadãos colaboram na criação da informação, surge a necessidade de existir uma aplicação para registar os dados adquiridos como, também analisar o estado de conservação de outras espécies animais e vegetais. Pretende-se então desenvolver uma aplicação *web* responsiva de forma a estar otimizada para equipamentos móveis tais como *Smartphones* e *Tablets* de forma a obter dados de cidadãos interessados em participar na recolha de informação, onde a área envolvida neste projecto é Portugal Continental e arquipélagos da Madeira e Açores.

1.2. Objectivo

O objectivo deste projecto foi desenvolver uma aplicação *web* responsiva que permite registar fotografias sobre espécies de estrelas-do-mar e outras informações através de um dispositivo móvel ou fixo. Cada utilizador pode publicar os dados na aplicação onde terá a sua conta associada. Posteriormente estes dados serão disponibilizados via *web* com uma informação mais detalhada como a visualização e descrição das fotografias, o autor e as localizações associadas a estes registos.

A aplicação utiliza a localização do dispositivo de cada utilizador. Por este meio, cada utilizador terá acesso a uma base de dados geográfica sobre o conteúdo de todos os registos existentes por parte de todos os utilizadores através de um mapa interactivo disponibilizado na aplicação.

De uma forma resumida o objectivo deste projecto é conceber uma aplicação *web* com as seguintes funcionalidades principais:

Funcionalidade 1 - Criar uma aplicação que registe fotografias através de um dispositivo móvel ou fixo.

Funcionalidade 2 - Associar o posicionamento a estes registos de forma dinâmica e visualizá-los através de um mapa interactivo.

Funcionalidade 3 - Disponibilizar toda a informação numa base de dados geográfica através de um mapa de acesso público aos utilizadores da aplicação.

Pretende-se também avaliar estas funcionalidades através de testes e aumentar a quantidade de informação com o aumento dos colaboradores.

1.3. Estado de Arte

No Estado de Arte deste projecto, a explicação tecnológica baseou-se em algumas aplicações já existentes que possuem algumas funcionalidades semelhantes, como por exemplo o *Facebook*¹ e o *Instagram*² onde é possível a georreferenciação de imagens associando uma localidade.

Além destas, existem outras também com funcionalidades bem semelhantes como a localização em tempo real de viaturas, bens e pessoas. Um destes casos é a aplicação GeoCar que faz a gestão de frotas automóvel e navegação, e a GeoMedicina que permite aos profissionais de saúde a monitorização de factores que contribuem para a saúde e doença das grávidas, uniformizar o acesso aos recursos disponíveis e promover a redução de risco na gravidez através de medidas de promoção de saúde. Estas aplicações utilizam a georreferenciação de eventos e são soluções criadas pela empresa *Gisgeo Information Systems*³, sediada no Porto. (GISGEO, 2017)

Apesar da existência de aplicações semelhantes, estas não possuem a funcionalidade de georreferenciação dinâmica onde o utilizador arrasta um marcador no mapa. Além disso, o conteúdo da aplicação é demasiado específico, sendo diferente destas aplicações.

1.4. A Instituição

O Centro Português de Actividades Subaquáticas⁴ (CPAS) é uma associação portuguesa dedicada a estruturar, incentivar e ensinar o mergulho. O CPAS foi fundado em 1953, está classificado como Instituição de Utilidade Pública sem fins lucrativos, e tem a sua sede em Lisboa, no Restelo.

¹ Facebook <https://www.facebook.com/>

² Instagram <https://www.instagram.com/>

³ Gisgeo Information Systems <http://www.gisgeo.pt/>

⁴ CPAS www.cpas.pt, www.facebook.com/pg/Centro-Português-de-Actividades-Subaquáticas-262016068615

É uma Associação de carácter Técnico, Cultural e Científico. A acção do CPAS ao longo de 50 anos de existência tem sido visível sobretudo na promoção e ensino das técnicas de penetração no mundo subaquático, na preservação do património histórico submerso, na promoção de conferências e exposições sobre temas ligados ao mar e no desenvolvimento de actividades que promovam o estudo e conservação de fauna e flora subaquáticas (CPAS, 2011).

Actualmente o CPAS mantém a sua actividade na área da Formação, nomeadamente de Mergulho Amador, Náutica de Recreio, Fotografia subaquática e outros cursos como Identificação de espécies marinhas. Na área da sensibilização, promove para os associados e restante comunidade, sessões e palestras direccionadas para os problemas ambientais marinhos e, de forma a melhor contribuir para a protecção do meio marinho, pretende efectuar estudos científicos que suportem a sua conservação (ex. projecto de estudo de gorgónias; campanha de preservação do cavalo marinho, etc.) (CPAS, 2011).

Os esforços dos associados e voluntários do CPAS, juntamente com meios financeiros que o CPAS realiza (ex. quotas dos seus associados, donativos, administração de cursos) e subsídios que adquire por parte de entidades privadas e apoios governamentais, revertem a favor de acções e actividades em prol do mergulho em Portugal e na protecção do meio marinho (CPAS, 2011).

1.5. Planeamento

Para atender aos objectivos propostos, foi necessário elaborar um plano estrutural bem detalhado de modo a cumprir com as regras e modelos de desenvolvimento de *software* bem como requisitos necessários para o cumprimento dos mesmos. Nesta secção apresenta-se na Tabela 1 o planeamento das tarefas realizadas durante a elaboração deste projecto. As tarefas estão organizadas por tarefas principais e secundárias (quando necessário) com data de início e fim.

#	Tarefa	Data de início	Data de fim
1	<i>Sea-Stars Watching</i> – Compente Aplicacional	2016-09-30	2017-05-26
1.1	Planeamento funcional do projecto	2016-09-30	2016-10-30
1.2	Estudo das tecnologias a usar no projecto		
1.3	Contexto e requisitos do projecto		
2	Desenho físico e funcional	2016-11-01	2016-11-16
2.1	Desenho e modelação da base de dados		
2.2	Desenho da arquitectura geral		
2.3	Definição do ambiente de desenvolvimento		

3	Desenvolvimento aplicacional e testes	2016-11-20	2017-05-28
3.1	Criação do ambiente de desenvolvimento		
3.2	Codificação de todos os componentes		
3.3	Testes		
4	Dissertação	2017-06-01	2017-07-25

Tabela 1 - Planeamento do projecto.

1.6. Notação Adoptada

A Língua escolhida para a escrita deste documento foi o Português de acordo com o antigo acordo ortográfico. Para bibliografia e citações foi utilizado o sistema padrão *Harvard*. Todo o conteúdo escrito noutro idioma está apresentado em itálico.

1.7. Estrutura da Dissertação

Este documento é constituído por cinco capítulos onde cada um apresenta diferentes conteúdos sobre o projecto e encontra-se organizado da seguinte forma:

Capítulo 1 - Este capítulo tem como finalidade apresentar e descrever a motivação e o objectivo. É apresentada também a instituição para o qual o projecto foi elaborado, o planeamento, notação adoptada e a estrutura da dissertação.

Capítulo 2 - É apresentado o contexto do projecto na medida em que são apresentadas as ideias conceptuais para o seu desenvolvimento. São discutidas as soluções adoptadas com base em tecnologias e ferramentas a utilizar, bem como soluções alternativas.

Capítulo 3 - Nesta secção é apresentada a estrutura conceptual do projecto desde, arquitectura geral, até à definição de todas as ferramentas e tecnologias usadas, ambiente de desenvolvimento, instalação e funcionamento geral da aplicação.

Capítulo 4 - São descritos todos os testes realizados e as conclusões sobre os mesmos.

Capítulo 5 - Neste capítulo são apresentadas as conclusões relativas ao projecto desenvolvido, estado actual do projecto, problemas e limitações encontradas, contribuições e perspectivas de trabalho futuro.

Capítulo 2 – Contexto do Projecto

2.1. Introdução

No capítulo anterior foi apresentado o planeamento do trabalho, a motivação e objectivos a cumprir, estado de arte, notação adoptada e estrutura da dissertação. Antes de prosseguir para o desenvolvimento, foi necessário realizar uma pesquisa sobre as tecnologias e ferramentas a utilizar. Após a pesquisa foram encontradas algumas soluções que envolvem plataformas tecnológicas de suporte à programação e desenvolvimento *web*. Este capítulo está dividido em duas partes:

Parte I - Descreve a solução adoptada em contraste com soluções alternativas.

Parte II - Descreve as soluções alternativas para o desenvolvimento deste projecto.

2.2. Solução Adoptada

A solução adoptada partiu da experiência em projectos anteriormente desenvolvidos e nas ferramentas e tecnologias usadas, de forma a cumprir com os objectivos definidos. Foram seleccionadas as melhores soluções para atingir o que estava proposto.

2.2.1. SGBD

O modelo de dados está assente numa estrutura em linguagem SQL, constituindo a linguagem adoptada para o desenho da base de dados da aplicação. Esta BD está assente numa plataforma *MySQL Server* (MySQL, 2017).

O *MySQL Server*⁵ é um SGBD de código aberto onde é possível que qualquer pessoa possa obter o *software* pela Internet e usá-lo sem qualquer pagamento. Foi originalmente desenvolvido para lidar com grandes bases de dados de uma forma muito mais rápida que as soluções existentes oferecendo também um conjunto de funcionalidades bastante útil e rico. A conectividade, velocidade e segurança tornam o *MySQL Server* muito recomendável para alojar e aceder a bases de dados na Internet (MySQL, 2017).

É um sistema de gestão de bases de dados relacional, que pode ser usado para guardar, gerir, actualizar e recolher dados (Couto, Francisco, 2011).

⁵ MySQL <https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html>

2.2.2. Servidor Web

Um servidor *Web* é uma aplicação informática responsável por aceitar pedidos HTTP e devolver as respectivas respostas que normalmente são documentos HTML ou objectos tais como imagens, documentos PDF, etc. Um dos mais populares servidores *Web* é o Apache que pode ser instalado em diversos sistemas operativos (Ver figura 1) (Couto, Francisco, 2011).

Para este projecto foi utilizado o servidor *Web* Apache uma vez que, além de suportar a linguagem PHP, é esta a linguagem de programação adoptada para este projecto.

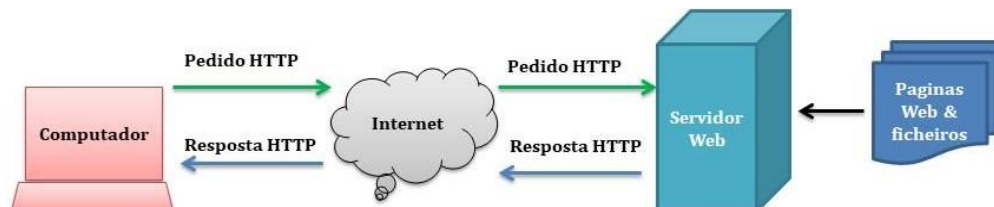


Figura 1 - Servidor Web.

2.2.3. Equipamento, Tecnologias e Ferramentas

A linguagem principal do projecto é PHP⁶. O PHP é uma linguagem de programação muito conhecida e direccionada para o desenvolvimento de aplicações *Web*, e que recorre a outras linguagens de programação de forma a disponibilizar todo o conteúdo vindo de uma Base de dados ou *Web Services*. É uma linguagem rápida e flexível permitindo a elaboração de uma aplicação *web* através de uma *framework* denominada Laravel 5.3.

Na Tabela 2 é apresentada a lista de tecnologias de programação e as ferramentas adoptadas nesta solução.

⁶ PHP <http://php.net>

Tecnologias/Ferramentas	Versão	Finalidade
PHP	7.0.9	Desenvolvimento <i>web</i>
HTML	5	Desenvolvimento <i>web</i>
CSS	3	Desenvolvimento <i>web design</i>
jQuery	3.1.1	Desenvolvimento <i>web/user interface</i>
JavaScript	ECMAScript 7, ECMAScript 6	Desenvolvimento <i>web/user interface</i>
JSON	2.0.1 – 2.1.0	Linguagem comunicação entre pedidos e respostas
MySQL	5.7	Base de dados
Apache	2.4.23	Servidor <i>web</i>
Laravel	5.3	<i>Framework</i> de desenvolvimento web
Postman	5.3.2	Testes de pedidos HTTP
XAMPP	7.0.9	Framework que contém Servidor <i>Web</i> e MySQL
Composer	N.A.	Instalação da <i>framework</i> Laravel
SQL	5.7	Desenvolvimento base de dados
Bootstrap	3.3.7	Framework de desenvolvimento <i>web design</i>

Tabela 2 - Tecnologias utilizadas.

Em termos de equipamento, foi utilizado um computador portátil com um processador *Intel Core i7*, 12GB de memória RAM e sistema operativo *Windows 10 Home* de 64 bits. Todo o *software* foi desenvolvido na linguagem principal PHP utilizada através da *framework* Laravel 5.3 onde foram também utilizadas outras linguagens fundamentais como HTML5, JavaScript, jQuery e CSS3. Nesta última recorre-se também à *framework* Bootstrap 3.3.7. Para poder instalar a *framework* Laravel foi necessário instalar uma ferramenta denominada Composer.

De seguida serão descritas algumas ferramentas/tecnologias que ainda não foram explicadas:

Bootstrap⁷- O Bootstrap é uma *framework* de *front-end* gratuita para desenvolvimento *Web* que torna o desenvolvimento mais fácil e rápido. Inclui modelos de *design* baseados em HTML e CSS para tipografia, formas, botões, tabelas, navegação, modais, carrosséis de imagens e muitos outros, bem como *plugins* de JavaScript opcionais. Além disto, tem a capacidade de criar projectos com *Design Web* responsivo. O *design Web* responsivo permite a criação de aplicações *web* onde o *design* é ajustado automaticamente às dimensões de todos os dispositivos desde *smartphones*, *tablets* a computadores portáteis ou fixos. (Bootstrap, 2017)

Laravel⁸- O Laravel 5.3 é uma *framework web* gratuita em PHP, criada por Taylor Otwell para desenvolvimento de aplicações *Web* através do um padrão de arquitectura *Model-View-Controller* (MVC). Algumas das características do Laravel são, uma linguagem simples e concisa, um sistema de gestão modular com um gestor de dependências dedicado, diferentes formas de

⁷ Bootstrap, <http://getbootstrap.com>

⁸ Laravel <https://laravel.com>

aceder a uma base de dados relacional e outras funcionalidades que ajudam na implementação e manutenção de aplicações. (Laravel, 2017)

Composer⁹ – O Composer é uma ferramenta de gestão de dependências de PHP que permite instalar as bibliotecas das quais um projecto depende. O Laravel utiliza o Composer para gerir as suas dependências. Desta forma, antes de poder usar o Laravel, é necessária a instalação desta ferramenta. (Composer, 2017)

XAMPP¹⁰ - É uma distribuição do Apache para os programadores que facilita a criação de um servidor *Web* local para fins de teste e alocação de aplicações. Contém tudo o que é necessário para configurar um servidor *Web* desde o próprio servidor (Apache *Server*), base de dados (MariaDB/MySQL) e linguagem de programação (PHP). O XAMPP é uma plataforma cruzada, o que significa que funciona em diferentes sistemas operativos como o Linux, Mac e Windows. Funciona como um simulador de um servidor *Web* público o que torna acessível a transição de um servidor de local para um servidor público. (XAMPP, 2017)

PhpMyAdmin¹¹ - O PhpMyAdmin é uma ferramenta de *software* livre feita em PHP com a finalidade de lidar com a gestão do MySQL na *Web*. O PhpMyAdmin suporta uma variada gama de operações no MySQL e no MariaDB. As operações mais utilizadas são a gestão da base de dados, criação e edição tabelas, colunas, relações, índices, utilizadores, permissões, entre outras. Estas operações podem ser executadas através da interface de utilizador que, ao mesmo tempo, tem a capacidade de executar qualquer declaração SQL. (PhpMyAdmin, 2003-2017)

Postman¹²- O Postman é uma aplicação que permite construir pedidos HTTP através de uma estrutura REST para testar serviços da *Web*. O Postman permite testar, desenvolver e documentar uma API e construir pedidos HTTP simples e complexos pelos utilizadores (Postman, 2017).

2.3. Soluções Alternativas

Nesta secção são abordadas outras possíveis soluções que podem constituir uma alternativa à solução actualmente implementada.

2.3.1. Alternativa ao SGBD

Além do MySQL existem outras possibilidades onde os dados podem ser armazenados. Serão descritas duas possibilidades em relação ao actual.

⁹ Composer <https://getcomposer.org>

¹⁰ XAMPP <https://www.apachefriends.org/about.html>

¹¹ PhpMyAdmin <https://www.phpmyadmin.net/>

¹² Postman <https://www.getpostman.com/>

Microsoft SQL Server¹³ - Tal como o MySQL, o Microsoft SQL Server é um SGBD relacional que suporta uma grande variedade de aplicações de processamento de transacções, análise e inteligência de negócio em sistemas de informação. É uma das três tecnologias de armazenamento de dados de topo no mercado. Esta, porém, está mais orientada a plataformas de Microsoft que utilizam uma linguagem de programação diferente como o C# e VB. NET. (SearchSQLServer, 2017)

Oracle Database¹⁴ - De igual forma como o *Microsoft SQL Server* e MySQL, a BD Oracle é também um SGBD relacional. Possui estruturas lógicas e estruturas físicas separadas. Devido a esta separação, o armazenamento físico dos dados pode ser gerido sem afectar o acesso a estruturas de armazenamento lógicas. A DB Oracle é um dos mecanismos de armazenamento de dados relacionais mais fiáveis e utilizados. (Oracle, 2017)

2.3.2. Alternativa ao Servidor Web

Em contraste com o Servidor *web* Apache, existem outras alternativas para o desenvolvimento do projecto. Serão apresentadas duas alternativas em relação ao Servidor *Web* actual.

Internet Information Services (IIS)¹⁵ - O IIS é um servidor *Web* flexível, seguro e manuseável com uma arquitectura em escala e aberta com o objectivo de armazenar dados ou aplicações *Web*. O IIS é mais orientado ao Windows onde são usados sistemas da Microsoft (SearchWindowsServer, 2008).

Apache Tomcat¹⁶ - O Tomcat é um servidor *Web* do *Apache Software Foundation*¹⁷ que disponibiliza páginas da *Web* que incluem a codificação JSP. O Tomcat é o resultado de uma colaboração entre programadores e está disponível no *site* Apache em versões binárias e de origem. Este servidor *web* é mais orientado a aplicações que utilizam a linguagem Java (Apache, 2017).

2.3.3. Alternativas de Tecnologias

Tal como o PHP, existem diversas opções que poderiam servir como alternativa ao desenvolvimento deste projecto. Essas alternativas podiam ser o uso da linguagem C#, Java, Python ou mesmo o Ruby por exemplo. A estas, estão associadas diferentes plataformas como a ASP.NET Framework, Spring Framework, Django Framework ou Ruby on Rails Framework.

¹³ Microsoft SQL Server, <http://searchsqlserver.techtarget.com/definition/SQL-Server>

¹⁴ Oracle Database https://docs.oracle.com/cd/B19306_01/server.102/b14220/intro.htm

¹⁵ IIS <https://www.iis.net/>

¹⁶ Apache Tomcat <http://tomcat.apache.org/>

¹⁷ Apache Software Foundation <https://www.apache.org/>

2.4. Sumário

Neste capítulo, foi apresentado o *software* adoptado para o desenvolvimento do projecto para o ambiente de desenvolvimento, servidor *web* e modelo de dados. Foram descritas as ferramentas e as tecnologias utilizadas para chegar ao resultado pretendido. Analisaram-se diversos documentos e bibliotecas que contribuíram para o mesmo. Foram também analisadas outras soluções que correspondem a soluções alternativas a nível de código, armazenamento de dados e servidores. No próximo capítulo são explicados os componentes escolhidos, bem como toda a arquitectura geral da aplicação.

Capítulo 3 – Trabalho realizado

Neste capítulo é apresentado o desenvolvimento do projecto desde as etapas, a equipa, a arquitectura geral, até ao ambiente de desenvolvimento onde foi desenvolvida a aplicação.

Para cumprir os objectivos do projecto e face ao facto de não existir uma equipa de trabalho, foi necessário implementar uma metodologia de trabalho bem definida e planeada de forma a garantir a excelente qualidade e rentabilidade do mesmo. Para que as datas serem cumpridas, a solução adoptada envolveu um nível tecnológico adaptado ao conhecimento actual na altura.

3.1. Metodologia e Equipa

Neste projecto, devido ao facto ser eu o único elemento responsável pelo desenvolvimento, foi muito importante o planeamento garantindo que tudo decorreria dentro do previsto. Este projecto foi supervisionado por mim, orientadores e pela responsável do projecto desde a sua raiz até à sua conclusão. O trabalho foi desenvolvido em casa e nos edifícios da Faculdade de Ciências da Universidade de Lisboa e toda a comunicação foi efectuada pessoalmente ou por correio electrónico com os respectivos orientadores.

A metodologia abordada envolve a experiência em projectos académicos realizados anteriormente. O modelo de desenvolvimento baseia-se no modelo em Cascata onde cada etapa constitui uma solução à medida. É caracterizada pelas etapas de Planeamento, Análise, Desenho, Análise, Desenvolvimento e Testes. Na Figura 2 são apresentadas as etapas envolvidas no projecto.

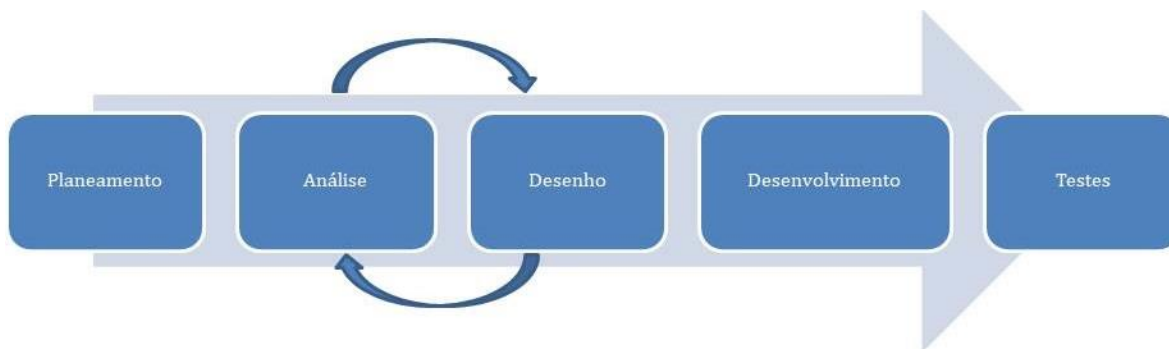


Figura 2 - Metodologia abordada no desenvolvimento do projecto.

O planeamento consistiu na construção num plano com metas e datas definidas para as várias etapas. Na primeira etapa de Análise, foram analisados os requisitos necessários para que a aplicação se concretizasse de forma a garantir que não existia falta de recursos para a execução do projecto. Na etapa de Desenho foi feito o desenho da BD da aplicação bem como a estrutura geral do funcionamento da aplicação. Na segunda etapa de Análise, foram analisados o desenho da BD e da estrutura da aplicação de forma averiguar se existia a necessidade de acrescentar ou retirar algum conteúdo tanto em termos técnicos como lógicos. Na etapa de Desenvolvimento procedeu-se à construção do projecto a nível de código e sistema. Finalmente, na etapa de Testes, a aplicação foi validada e testada através de testes e pelo responsável do projecto.

3.2. Arquitectura Geral

A arquitectura geral é constituída por duas camadas: *Front-End* e *Back-End*. Estas camadas descrevem o funcionamento lógico do *software*, variáveis e os componentes envolvidos, conjunto de bibliotecas, estrutura da base de dados, componentes lógicos executáveis e outros que compõem fisicamente o *software*.

3.2.1. Front-End e Back-End

Na Figura 3 apresentam-se estas duas camadas que compõem a arquitectura.

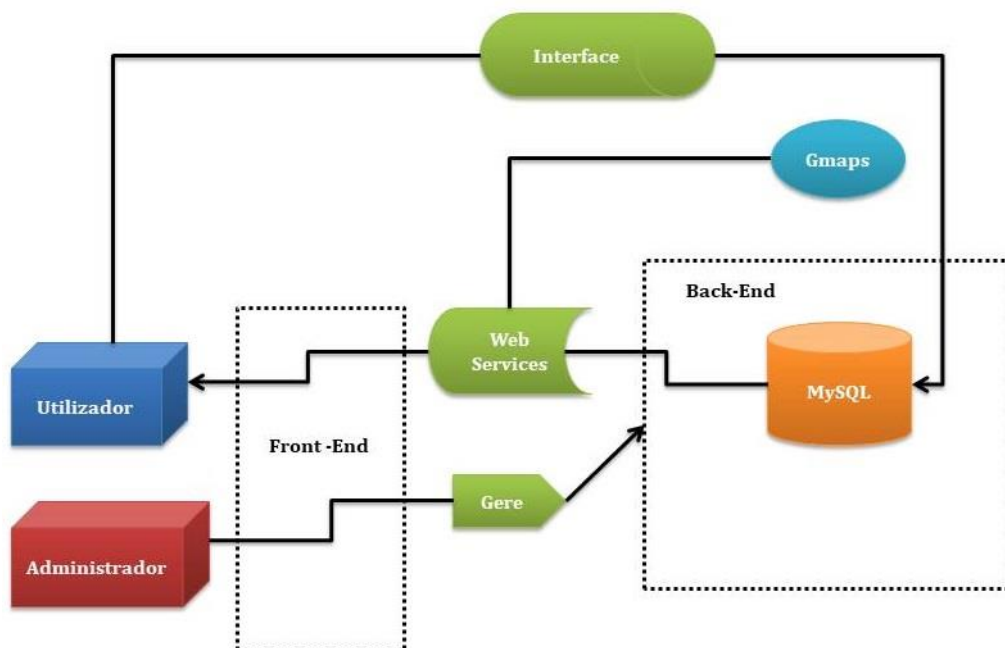


Figura 3 - Arquitectura geral da aplicação.

Front-End - O *Front-End* é o termo utilizado para caracterizar a interface entre o utilizador e a aplicação. Por outras palavras, é tudo o que envolve a visualização do utilizador (Desenho da aplicação/*Web-Design*). (Pluralsight, 2017).

Back-End - O *Back-End* é o termo utilizado para caracterizar o lado do servidor, ou seja, como a aplicação trabalha. É tudo o que o utilizador não vê no *browser*. É o que faz com que o *Front-End* seja possível. Consiste num servidor, uma aplicação e uma base de dados. É onde se constrói e é mantida a tecnologia que alimenta todos estes componentes, que quando juntos, permitem a existência do lado do utilizador (*Front-End*) e para que a aplicação exista (Pluralsight, 2017).

Web Services - Um *Web Service* é um termo genérico usado para uma funcionalidade de *software* que está alocada numa rede com um endereço de localização específico. O termo *Web Service* também pode ser com um significado mais específico como um serviço baseado em SOAP, que é descrito usando um documento WSDL. Na aplicação são usados serviços baseados em JSON (Tutorials Point, 2017).

Os *Web Services* baseados em JSON são diferentes dos serviços baseados em SOAP pelas razões que se apresentam:

- O conteúdo de uma mensagem SOAP contém dados XML, enquanto uma mensagem JSON contém dados JSON (Tutorials Point, 2017).
- O JSON e XML têm uma codificação diferente para descrever a estrutura dos dados. O JSON é mais eficiente e leve, logo uma mensagem JSON tipicamente será menor do que uma mensagem XML equivalente (Tutorials Point, 2017).
- O SOAP fornece um mecanismo para adicionar Cabeçalhos a uma mensagem e uma família de especificações para qualidades de serviço (como configuração de segurança e transações distribuídas). O JSON não fornece esse mecanismo e, em vez disso, depende dos serviços do protocolo de rede HTTP subjacente (Tutorials Point, 2017).

3.2.2. Esquema de Dados

O modelo de dados (Ver Figura 4) representa os dados da aplicação, regras de negócios, lógica e funcionalidades. É constituído por seis entidades que resultam em seis tabelas de dados. Na figura seguinte é apresentado a estrutura do modelo de dados em forma de um esquema com uma estrutura UML.

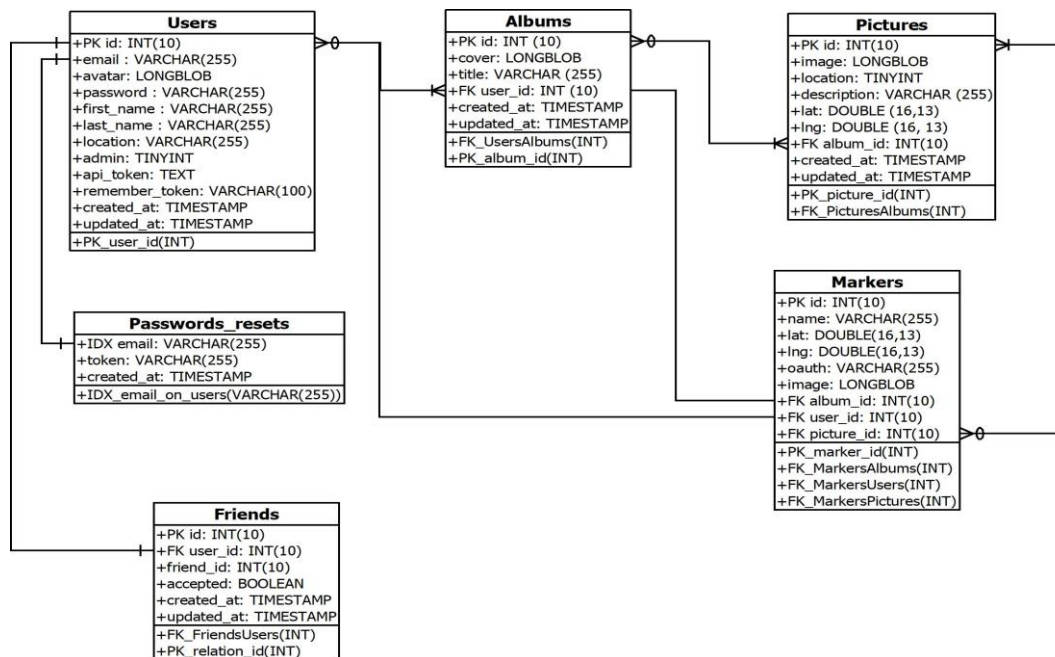


Figura 4 - Esquema UML do Modelo de dados.

No modelo de dados temos entidades, classes e objectos. Da análise da Figura 4 podemos verificar a lógica existente do modelo de dados e que é descrita da seguinte forma:

Users - Tabela que descreve e armazena os registos dos utilizadores da aplicação e esta relaciona-se com mais quatro tabelas: *Albums*, *Friends*, *Markers* e *Passwords_resets* através das chaves estrangeiras *user_id* e *email*.

Albums - Tabela que descreve e armazena um álbum de fotografias. O utilizador pode criar uma ou mais álbuns onde irá colocar as fotografias e os dados inerentes. Esta relaciona-se com mais três tabelas: *Users*, *Pictures* e *Markers* através das chaves estrangeiras *user_id* e *album_id* respectivamente.

Pictures - Tabela que descreve e armazena a fotografia que o utilizador coloca nos álbuns. Esta relaciona-se com duas tabelas: *Markers* e *Albums* através das chaves estrangeiras *picture_id* e *album_id*.

Friends - descreve e armazena os registos das relações entre utilizadores da aplicação. Esta relaciona-se com a tabela *Users* através da chave estrangeira *user_id*.

Markers - Tabela que armazena os registos dos pontos georreferenciados. Estes pontos estão associados às fotografias que por sua vez se associam aos álbuns dos utilizadores. Relaciona-se com mais três tabelas: *Users*, *Pictures* e *Albums* através das chaves estrangeiras *user_id*, *picture_id* e *album_id* respectivamente.

Passwords_resets - Tabela que armazena os *emails* e os *access tokens* dos utilizadores para recuperar as *passwords* das suas contas. Esta relaciona-se com a tabela *Users* através do *email* que funciona como chave estrangeira.

3.2.3. Componente Lógica de Desenvolvimento

O ambiente de desenvolvimento está assente na *framework* Laravel com estrutura *Model-View-Controller* com o acrónimo MVC¹⁸.

O padrão de arquitectura MVC separa uma aplicação em três componentes principais: o Modelo, a Vista e o Controlador. O MVC isola a lógica da aplicação da camada de interface de utilizador favorecendo a organização da estrutura do código. O Controlador recebe todos os pedidos feitos pelo utilizador e seguidamente trabalha com o Modelo para preparar os dados necessários para a Vista. A Vista usa os dados preparados pelo Controlador para gerar uma resposta final que seja apresentável. (Microsoft, 2017)

A arquitectura MVC pode ser representada graficamente na Figura 5.

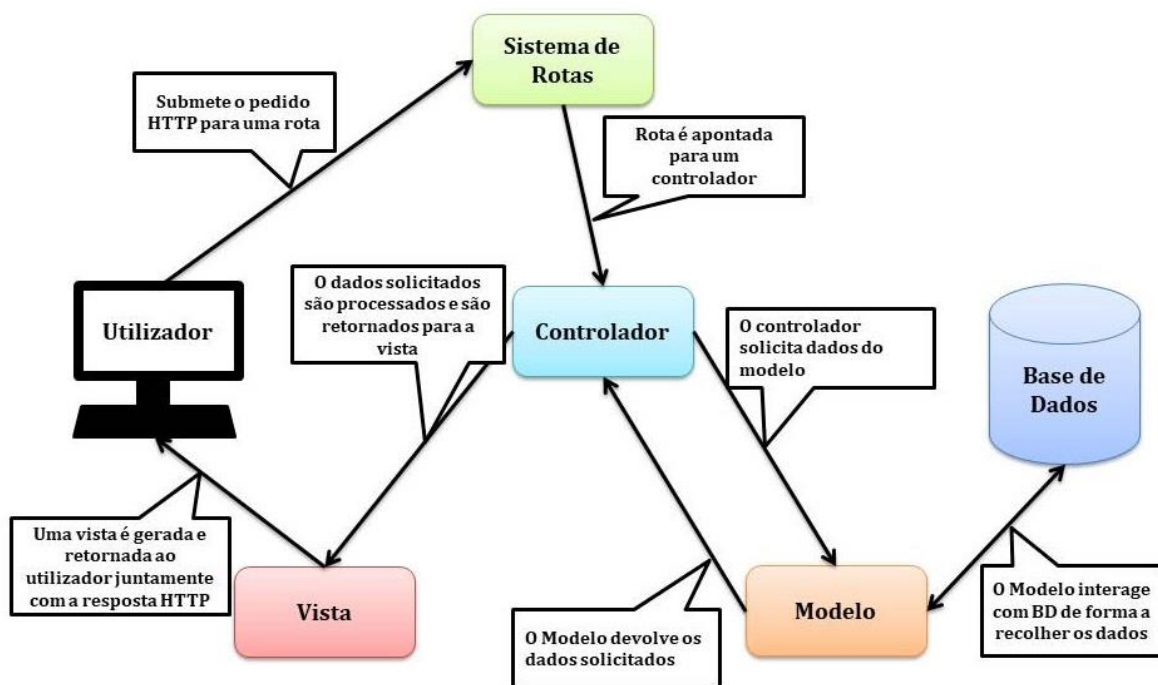


Figura 5 - Padrão de arquitectura MVC.

¹⁸ MVC https://www.tutorialspoint.com/struts_2/basic_mvc_architecture.htm

Os três componentes constituem a base desta arquitectura padrão e podem ser explicados da seguinte forma:

Modelo - O modelo é responsável pela gestão dos dados da aplicação. Responde ao pedido que vem do utilizador através da Vista e também responde aos comandos do controlador para actualização e recolha de dados. Nesta aplicação existem quatro modelos: *User*, *Album*, *Picture* e *Marker*.

Vista - A Vista é a componente responsável pela apresentação de dados num dado formato específico, retornada pelo controlador. São páginas baseadas em HTML e em *scripts* como o JSP, ASP, PHP e muito fáceis de integrar com a tecnologia AJAX.

Controlador - O Controlador é responsável por dar resposta ao pedido HTTP do utilizador e interagir com os objectos do Modelo de dados. O Controlador recebe o pedido HTTP, valida o pedido e de seguida executa o comando que modifica o estado do modelo de dados. A aplicação é constituída por doze controladores todos com fins distintos. Na Tabela 3 é apresentada a lista dos controladores da aplicação.

Controlador	Função
RegisterController	Controla o registo de novos utilizadores, bem como sua validação e criação.
LoginController	Controla e trata da autenticação dos utilizadores da aplicação e redireccionando-os para a página inicial.
ResetPasswordController	Responsável pelo tratamento de pedidos de redefinição de <i>passwords</i> .
ForgotPasswordController	Responsável pelo tratamento de e-mails para redefinição de <i>passwords</i> .
HomeController	Controla o redireccionamento de utilizadores após a validação processo de autenticação.
ProfileController	Controla as páginas pessoais de cada utilizador. Permite a visualização dos perfis e alteração dos perfis do utilizador autenticado.
SearchController	Faz o controlo de todos os pedidos de pesquisa de utilizadores por outros utilizadores.
PictureController	Controla os pedidos que permitem a visualização, actualização, adição e remoção de fotografias.
AlbumController	Controla os pedidos que permitem a visualização, actualização, adição e remoção dos álbuns dos utilizadores
FriendsController	Controlador responsável pelos pedidos que alteram o estado da relação entre utilizadores.
MarkersController	Controla os pedidos que permitem a visualização de todos os pontos no mapa interactivo.
MailController	Responsável pelo envio de <i>emails</i> por parte dos utilizadores à equipa técnica da aplicação.

Tabela 3 - Controladores da aplicação.

Sistema de Rotas - O sistema de rotas define o padrão de URL e as informações do controlador. Todas as rotas configuradas de uma aplicação estão armazenadas num ficheiro PHP e serão usadas pelo sistema para determinar a classe ou Controlador apropriado para um dado pedido HTTP. Posteriormente será explicada de forma mais detalhada a constituição deste sistema (Microsoft, 2008).

3.2.4. Estrutura Rest

A estrutura REST, denominada como *REpresentational State Transfer* (REST) é um padrão de *design* de software usado para aplicações *web*. A ideia básica do REST é tratar objectos no lado do servidor como recursos que podem ser criados, actualizados ou destruídos. É uma forma de formatar as URIs nas aplicações *web* (Oracle, 2013).

Este padrão envolve quatro métodos e quatro operações básicas denominadas por um nome simbólico – CRUD (Tutorials Point, 2017).

Método	Operação
POST	Create
GET	Read
PUT/PATCH	Update
DELETE	Delete

Tabela 4 - Métodos e operações CRUD.

Estes métodos são denominados por HTTP. Estes métodos definem pedidos que indicam uma operação/acção que pode ser de leitura (GET), criação (POST), actualização (PUT/PATCH) e de remoção (DELETE).

Na estrutura REST, o servidor fornece acesso a recursos onde o cliente, neste caso o utilizador, acede e visualiza os recursos. Cada recurso é identificado por URIs ou indentificadores globais. A arquitectura REST usa várias representações para representar um recurso como simples texto, JSON e XML. O JSON é o formato que tem vindo a ser mais usado em *web Services* (Tutorials Point, 2017).

Utilizando a *framework* Laravel é facilmente disponibilizada a informação através de um *Web Service*, que recolhe os dados da Base de dados e os passa para os controladores em JSON e daí para a Vista onde são transformados para serem visualizados no ecrã. Os dados são transformados através de um engenho do Laravel denominado de *blade.php* que permite a visualização dos dados JSON.

As tabelas seguintes apresentam a estrutura dos recursos REST da aplicação e as tabelas estão apresentadas pelo nome do recurso, método, parâmetros recebidos e os resultados esperados.

Na Tabela 5 são descritos os recursos gerais disponíveis, isto é, os recursos que tanto um utilizador autenticado como um não autenticado têm acesso. Esta estrutura interage com envios de *email* para a administração, página sobre a aplicação e de boas vindas.

Recurso	Método	Parâmetros	Resultados
/	GET		Devolve a página inicial da aplicação
/about	GET		Devolve a página que explica o que é a Aplicação.
/mail	GET		Devolve a página para enviar <i>email</i> a administração da Aplicação.
/mail/post	POST		Submissão de <i>email</i> para administração.

Tabela 5 - Estrutura REST de recursos gerais.

Na Tabela 6 são descritos os recursos disponíveis para autenticação e registo de utilizadores. Estes recursos permitem a criação de contas de utilizador na aplicação bem como as suas validações.

Recurso	Método	Parâmetros	Resultados
/login	GET		Devolve a página de autenticação de utilizadores.
/login	POST		Submissão da autenticação dos utilizadores que poderá ser validada ou rejeitada.
/register	GET		Devolve a página de registo de utilizadores.
/register	POST		Submissão do registo dos utilizadores que na aplicação.
/logout	POST		Desliga a sessão do utilizador autenticado.

Tabela 6 - Estrutura REST para autenticação e registo de utilizadores.

Na Tabela 7 são apresentados os recursos REST para a manutenção da conta do utilizador. Estes recursos envolvem um mecanismo de renovação de *password* do utilizador e enviam para o e-mail do mesmo um endereço com um recurso para fazer uma *password* nova.

Recurso	Método	Parâmetros	Resultados
/password/reset	GET		Devolve a página para pedir <i>reset</i> de <i>passwords</i> de utilizadores.
/password/email	POST		Submete o envio do endereço para o <i>reset</i> da <i>password</i> para email do utilizador.
/password/reset/{token}	GET	<i>token</i> = id da sessão	Devolve a página com um formulário para o utilizador escrever uma nova <i>password</i> .
/password/reset	POST		Submissão da nova <i>password</i> do utilizador.

Tabela 7 - Estrutura REST para renovação de *password*.

Na Tabela 8 é apresentada a estrutura dos serviços para interacção com os perfis dos utilizadores. Esta estrutura permite a visualização dos perfis dos utilizadores autenticados e não autenticados e a modificação do perfil no caso de ser autenticado.

Recurso	Método	Parâmetros	Resultados
/myprofile/{id}	GET	id = id do utilizador	Devolve o perfil do utilizador autenticado.
/userprofile/{id}	GET	id = id do utilizador	Devolve o perfil de um utilizador com um determinado id.
/myprofile/settings	GET		Devolve as configurações do perfil do utilizador autenticado.
/myprofile/settings/update/{id}	PUT	id = id do utilizador.	Actualização do perfil do utilizador.
/myprofile/settings/picture/{avatar}	PUT	avatar = fotografia de perfil	Actualização da foto de perfil do utilizador

Tabela 8 - Estrutura REST para alteração e visualização de perfis.

Na Tabela 9 são expostos os recursos disponíveis para a gestão dos álbuns de cada utilizador. Estes recursos permitem a visualização dos álbuns de utilizadores bem como a sua edição, criação e remoção.

Recurso	Método	Parâmetros	Resultados
/albums/myalbums	GET		Devolve todos os álbuns do utilizador autenticado.
/albums/useralbums/{id}	GET	id = id do utilizador	Devolve a página com dados que correspondem a todos os álbuns de um utilizador.
/albums/myalbums/post	POST		Submete a criação de um álbum por parte de um utilizador.
/albums/myalbums/edit/{album_id}	PUT	album_id = id do álbum	Actualiza um álbum de um utilizador.
/myalbums/delete/{album_id}	DELETE	album_id = id do álbum	Remove um álbum de um utilizador.

Tabela 9 - Estrutura REST para gestão de álbuns de utilizadores.

Na Tabela 10 estão representados os recursos disponíveis para interacção com as fotografias nos álbuns dos utilizadores. Estes recursos permitem a visualização, edição, criação e remoção de fotografias pertencentes aos álbuns.

Recurso	Método	Parâmetros	Resultados
/albums/myalbums/{id}/pictures	GET	id = id do álbum	Devolve a página com dados que correspondem a todas as fotografias existentes num álbum de um utilizador autenticado.
/albums/useralbums/{id}/pictures	GET	id = id do álbum	Devolve a página com dados que correspondem a todas as fotografias existentes num álbum de um utilizador.
/pictures/{picture_id}	GET	picture_id = id da fotografia	. Devolve uma fotografia com um determinado id.
/albums/myalbums/{id}/pictures/post	POST		Adiciona uma fotografia a um determinado álbum.
/albums/myalbums/{id}/pictures/{picture_id}/ /description	PUT	picture_id = id da fotografia, id = id do álbum	Actualiza a descrição de uma fotografia de um determinado álbum.
/albums/myalbums/{id}/pictures/{picture_id}/ /location	PUT	picture_id = id da fotografia, id = id do álbum	Actualiza a localização de uma fotografia de um determinado álbum.
/pictures/delete/{picture_id}	DELETE	picture_id = id da fotografia,	Remove uma fotografia de um álbum.

Tabela 10 - Estrutura REST para gestão de fotografias.

Na Tabela 11 encontram os recursos que permitem a interacção entre utilizadores. Estes recursos permitem a um utilizador adicionar outros utilizadores como amigos bem como a pesquisa por outros utilizadores na aplicação e consulta dos amigos já adicionados.

Recurso	Método	Parâmetros	Resultados
/friends	GET		Devolve a página com os amigos de um determinado utilizador.
/friends/add/{friend_id}	GET	friend_id = id do utilizador a adicionar	Envia um pedido de amizade por parte de um utilizador a outro utilizador.
/friends/accept/{friend_id}	GET	friend_id = id do utilizador de onde vem o pedido de amizade	Aceitar o pedido de amizade vindo de um utilizador.
/search/{string}	GET	string = parâmetro de pesquisa que pode ser uma letra ou palavras.	Devolve a página com os utilizadores no contexto do parâmetro pesquisado.

Tabela 11 - Estrutura REST para pesquisa e interligação de utilizadores.

Na Tabela 12 é apresentado o recurso que devolve todas as localizações das fotografias. Este recurso, além da georreferenciação traz também toda a informação inerente como a fotografia e autor associados o que permite além da visualização do ponto no mapa, o conteúdo fotográfico e o utilizador.

Recurso	Método	Parâmetros	Resultados
/markers/search	GET	Pedido HTTP com um intervalo de latitude e longitude.	Devolve todos os pontos georreferenciados e as fotografias e utilizadores inerentes.

Tabela 12 - Estrutura REST para recolha das fotografias georreferenciadas.

O recurso depois de chamado, retorna uma página e os dados em JSON como resposta ao pedido para depois injectar na página devolvida. Nas Figuras 7 e 8 é demonstrado um exemplo de uma resposta HTTP utilizando o recurso que devolve a página de perfil de um utilizador.

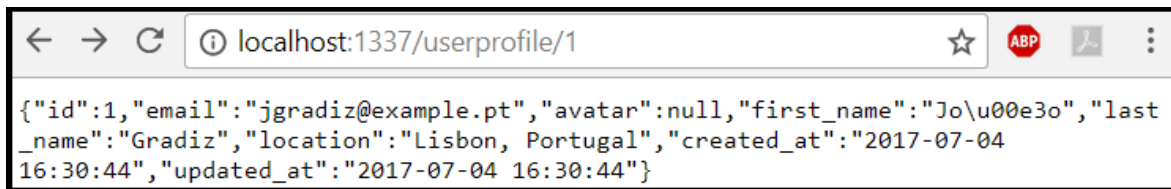


Figura 6- Resposta HTTP em JSON com os dados de um utilizador.

Após a resposta do recurso, esta informação é injectada na página e este conjunto é retornado para o ecrã onde é disponibilizado.

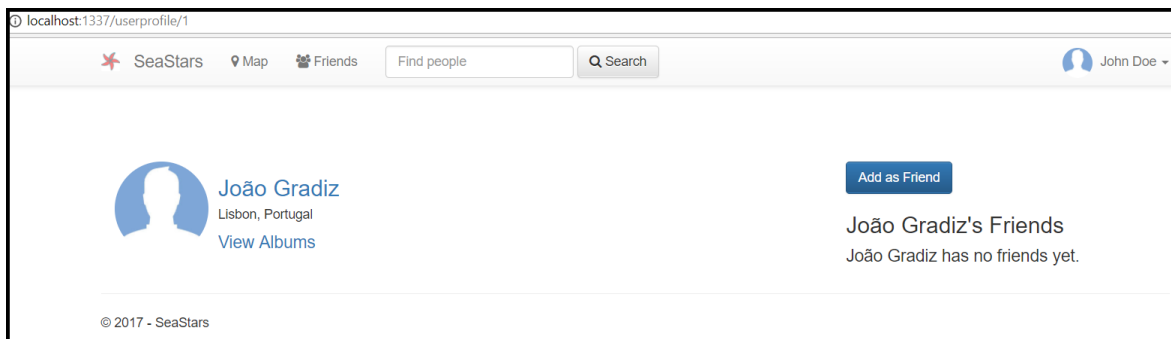


Figura 7 - Página devolvida com a informação do Utilizador.

3.2.5. Middleware

O sistema *Middleware* é um mecanismo de *software* que permite a filtragem de pedidos HTTP que acedem a aplicação. Este sistema actua como intermediário entre o pedido e a resposta. Por exemplo, o Laravel inclui um sistema que verifica se o utilizador da aplicação está autenticado ou não. Se o utilizador for autenticado, este será redireccionado para a página inicial; caso contrário, será redireccionado para a página de autenticação. Nesta aplicação são utilizados dois sistemas Middleware que advêm da instalação do projecto: Auth e *Web/Guest* (Laravel, 2017).

Middleware Auth - Este sistema é responsável por garantir o acesso e a passagem de pedidos HTTP a rotas estabelecidas na estrutura REST apenas se o utilizador estiver autenticado. Caso contrário, o utilizador fica impedido de entrar por essa rota e é retornado para a página de *login*.

Middleware Web/Guest - Este sistema garante o acesso e a passagem de pedidos HTTP a rotas estabelecidas na estrutura REST para utilizadores não autenticados. Assim, qualquer utilizador que não tenha uma conta associada tem autorização para aceder a estas rotas mesmo que não tenha uma conta associada. Por exemplo, a página de Autenticação, é uma página de acesso geral.

Na Figura 9 é apresentada a estrutura *Middleware*. Esta estrutura apresenta as duas camadas: Autenticados e Não Autenticados. Deste modo é garantida a autorização dos utilizadores a navegarem na aplicação sobre um determinado espectro de rotas sendo redireccionados para trás caso não tenham autorização para fazer o pedido para essa rota.

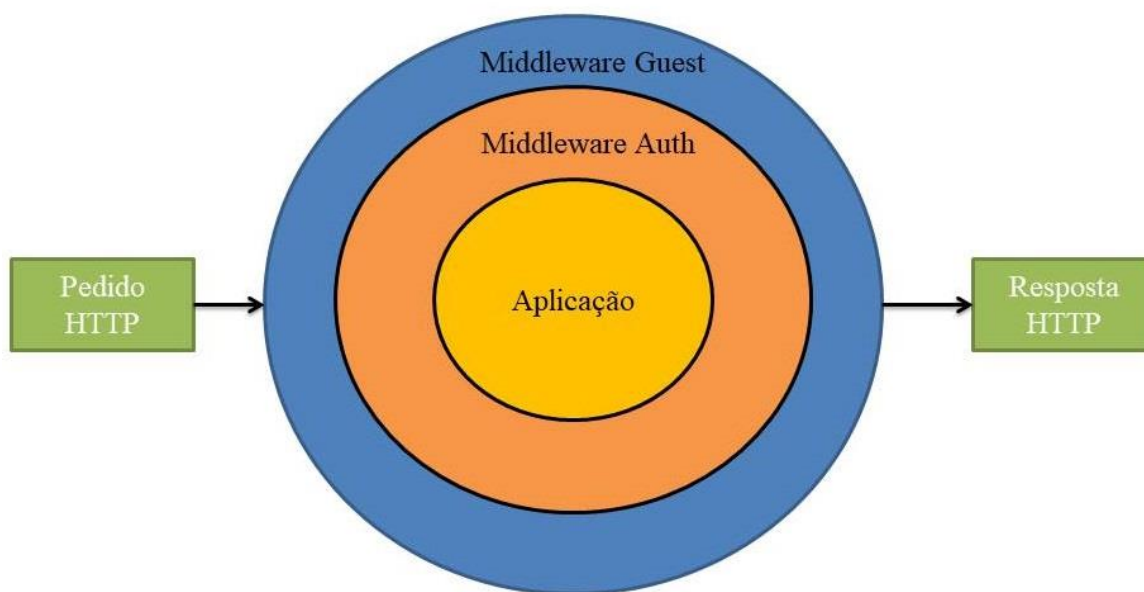


Figura 8 – Camadas do Sistema Middleware.

Além destes grupos de *Middleware* existem outros que podem ser utilizados dependendo da finalidade e uso da aplicação, e podem ser encontrados na aplicação no ficheiro *Kernel.php* e são apresentados na Figura10.

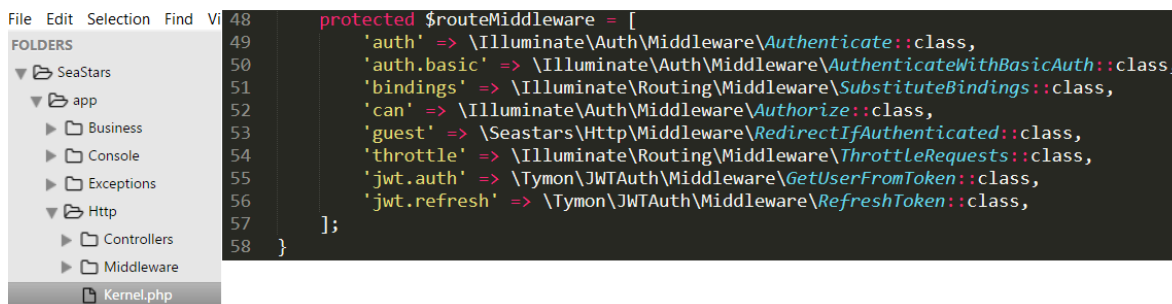


Figura 9 – Todas as camadas do Sistema Middleware.

Estes sistemas são utilizados num ficheiro dentro da aplicação denominado *web.php*. É onde é imposta a regra de redireccionamento e estabelecimento de rotas consoante o grupo que o utilizador pertence. Apresenta-se na Figura 11 o contexto de utilização destes recursos nalgumas rotas da aplicação.



Figura 10 - Aplicação do *Middleware* no sistema de rotas.

3.3. Ambiente de Desenvolvimento

Nesta secção é abordado o ambiente onde assenta o projecto. É descrito o processo de instalação da *framework* Laravel no servidor local, configuração da máquina virtual, comandos específicos da *framework*, ligação à base de dados, migração dos modelos para a base de dados MySQL e o funcionamento da aplicação.

3.3.1. Instalação do Ambiente de Desenvolvimento

Após reunir todas as condições para a criação da aplicação, procedeu-se a instalação do projecto com a *framework* Laravel utilizando o gestor de dependências *Composer*. A plataforma XAMPP contém uma directoria onde estão guardados os projectos que exijam conexão com servidor local. Através da linha de comandos do *Windows* é necessário localizar esta pasta e correr um comando específico: *composer create-project laravel/laravel <nome_do_projecto>*. Depois de correr o comando e a aplicação instalada, esta fica alocada na directoria com o nome do projecto e o ambiente Laravel fica pronto para começar os próximos desenvolvimentos. Nas próximas figuras (Figuras 12 e 13) são apresentados exemplos de execução dos comandos para uma aplicação e o conteúdo depois de instalado (Laravel, 2017).

```
C:\xampp\htdocs>composer create-project laravel/laravel AminhaPrimeiraApp

laravel/framework suggests installing league/flysystem-rackspace (Required to use the Nexmo client)
laravel/framework suggests installing nexmo/client (Required to use the Nexmo client)
laravel/framework suggests installing pda/pheanstalk (Required to use the Redis client)
laravel/framework suggests installing predis/predis (Required to use the Redis client)
laravel/framework suggests installing pusher/pusher-php-server (Required to use the Pusher client)
laravel/framework suggests installing symfony/dom-crawler (Required to use the Symfony client)
laravel/framework suggests installing symfony/psr-http-message-bridge (Required to use the Symfony client)
sebastian/global-state suggests installing ext-uopz (*)
phpunit/phpunit-mock-objects suggests installing ext-soap (*)
phpunit/php-code-coverage suggests installing ext-xdebug (^2.5.1)
phpunit/phpunit suggests installing phpunit/php-invoker (~1.1)
phpunit/phpunit suggests installing ext-xdebug (*)
Writing lock file
Generating autoload files
> Illuminate\Foundation\ComposerScripts::postUpdate
> php artisan optimize
Generating optimized class loader
The compiled services file has been removed.
> php artisan key:generate
Application key [base64:oyxwrAsdDb7JCxzfyZCkg6HlfQwnB5I6+JrUm6sU384=] set successfully.
C:\xampp\htdocs>
```

Figura 11 - Instalação da aplicação via Composer.

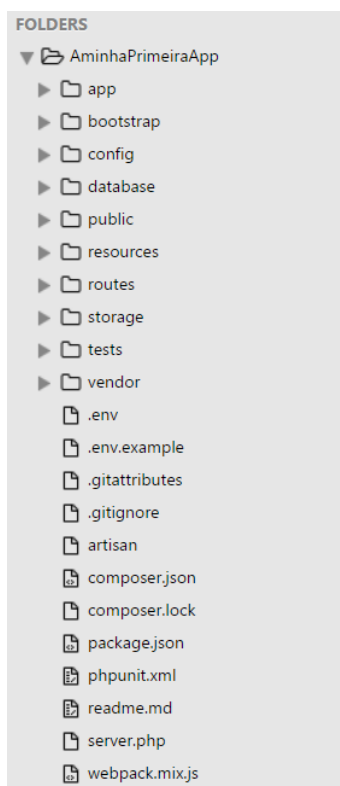


Figura 12 – Pastas e ficheiros da aplicação.

3.3.2. Configuração da Máquina Virtual

A máquina virtual é onde está alocado o servidor local. Esta armazena, torna visível e permite o funcionamento da aplicação *online*. É necessária a configuração da máquina virtual de forma a permitir o uso da aplicação através de um nome que é normalmente associado ao endereço ou IP da máquina. A máquina utilizada foi o próprio computador portátil e existiu a necessidade de configurar o servidor local de forma a simular um servidor público.

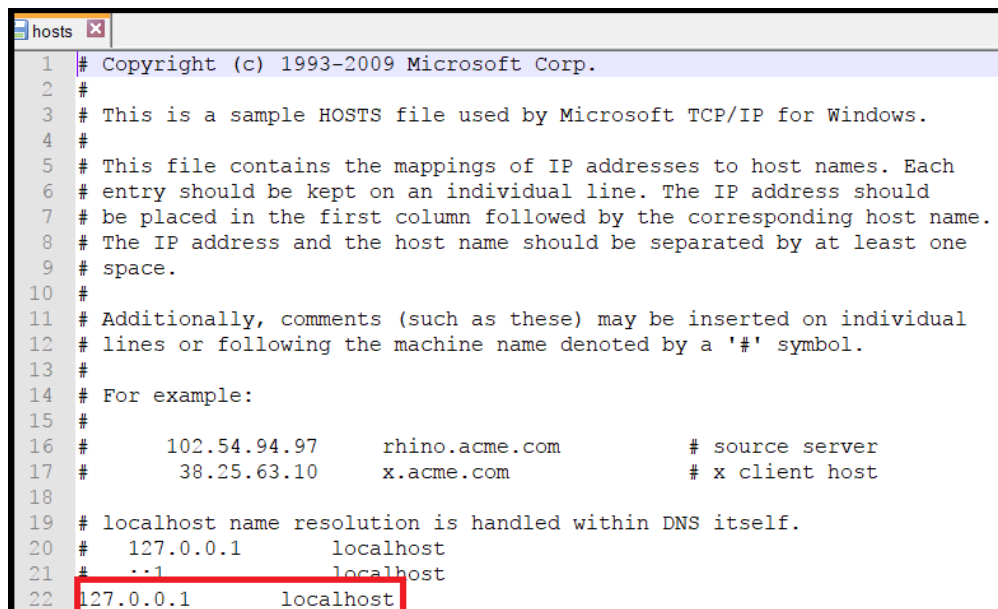
Para configurar a máquina virtual foi necessário configurar o servidor Apache para que aponte para pasta que contém ficheiros que representam o motor do projecto. De forma a cumprir com este objectivo foi necessário entrar no ficheiro de configuração de máquinas virtuais do Apache denominado por *httpd-vhosts.conf* e apontar para o motor da aplicação como se pode ver na figura seguinte (Figura 14).



```
32     ##ErrorLog "logs/dummy-host.example.com-error.log"
33     ##CustomLog "logs/dummy-host.example.com-access.log" combined
34 ##</VirtualHost>
35
36 ##<VirtualHost *:80>
37     ##ServerAdmin webmaster@dummy-host2.example.com
38     ##DocumentRoot "C:/xampp/htdocs/dummy-host2.example.com"
39     ##ServerName dummy-host2.example.com
40     ##ErrorLog "logs/dummy-host2.example.com-error.log"
41     ##CustomLog "logs/dummy-host2.example.com-access.log" combined
42 ##</VirtualHost>
43 ##NameVirtualHost *:80
44 <Directory C:/xampp/htdocs/SeaStars/public>
45     AllowOverride All
46     Require all granted
47 </Directory>
48 NameVirtualHost *:1337
49 <VirtualHost *:1337>
50     DocumentRoot "C:/xampp/htdocs/SeaStars/public"
51     ServerAlias SeaStars
52     ServerName SeaStars
53 </VirtualHost>
```

Figura 13 - Configuração da máquina virtual.

Esta máquina tem o nome de localhost:1337 ao qual está associado um IP específico que depende da localização onde esta se encontra. Como tal, este nome tem que estar associado ao motor do projecto que está no ficheiro *index.php*. Este ficheiro activa todas as dependências e módulos aplicativos de forma sequencial para garantir o funcionamento correcto. Além disto, foi também necessário configurar o sistema para esta virtualização de modo que o computador possa reconhecer e validar a máquina virtual através do mapeamento do IP. Este mapeamento associa o IP fixo do computador ao nome *localhost* (Ver Figura 15).



```
1 # Copyright (c) 1993-2009 Microsoft Corp.
2 #
3 # This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
4 #
5 # This file contains the mappings of IP addresses to host names. Each
6 # entry should be kept on an individual line. The IP address should
7 # be placed in the first column followed by the corresponding host name.
8 # The IP address and the host name should be separated by at least one
9 # space.
10 #
11 # Additionally, comments (such as these) may be inserted on individual
12 # lines or following the machine name denoted by a '#' symbol.
13 #
14 # For example:
15 #
16 #       102.54.94.97       rhino.acme.com       # source server
17 #       38.25.63.10       x.acme.com           # x client host
18 #
19 # localhost name resolution is handled within DNS itself.
20 #   127.0.0.1       localhost
21 #   ::1             localhost
22 127.0.0.1       localhost
```

Figura 14 - Mapeamento da máquina virtual.

3.3.3. Comandos Específicos do Laravel

O Laravel possui diversos comandos que funcionam como geradores de componentes. Estes podem ser modelos, migrações, controladores, entre outros. Nesta aplicação foram usados alguns destes comandos que envolvem os modelos, controladores, migração de tabelas de dados para BD e a simulação de um servidor local e público. (Laravel, 2017)

Na Tabela 13 seguinte é apresentada a lista de comandos usados durante o desenvolvimento.

Comando	Descrição
<i>php artisan make:controller</i> <nome_do_controlador>	Gerador de um controlador com um dado nome.
<i>php artisan make:model</i> <nome_do_modelo>	Gerador de um modelo com um dado nome.
<i>php artisan make:migration</i> <nome_da_migração>	Gerador de uma classe que representa a tabela de dados que irá estar na base de dados após migração.
<i>php artisan make:auth</i>	Gerador de controladores e vistas para autenticação e manutenção da conta dos utilizadores.
<i>php artisan migrate</i>	Migração de todas as classes que representam as tabelas de dados.
<i>php artisan migrate:reset</i>	<i>Rollback</i> de todas as migrações.
<i>php artisan migrate:rollback</i>	<i>Rollback</i> à ultima migração efectuada.
<i>php artisan serve</i>	Comando que possibilita a ligação a um servidor remoto.

Tabela 13 - Lista de comandos mais utilizados.

Todos os controladores, modelos e migrações também podem ser criados de forma manual, embora seja bastante mais rápido e eficaz através dos comandos da *framework* Laravel. Por exemplo, para criar o modelo *User* recorreu-se ao comando *php artisan make:model User* e para criar o controlador *HomeController* foi utilizado o comando *php artisan make:controller HomeController*. Os Controladores de autenticação *RegisterController*, *AuthController*, *ResetPasswordController* e *ForgotPasswordController* foram criados com o comando *php artisan make:auth*.

Embora seja muito mais rápido e prático, estes comandos apenas geram o esqueleto do código. Toda a lógica e desenvolvimento é posteriormente implementado.

3.3.4. Ligação à Base de Dados

A Ligação da aplicação à BD MySQL é parametrizada em dois ficheiros de configuração existentes dentro da própria aplicação que são os ficheiros *database.php* e o *.env*. No primeiro é configurado o driver de configuração que pode ser o MySQL, SQLite, PostgreSQL e outros mais (Figura 17). Este recolhe variáveis do segundo ficheiro onde estão guardadas as credenciais de acesso permitindo assim a conexão de forma correcta à BD (Figuras 16 e 17).

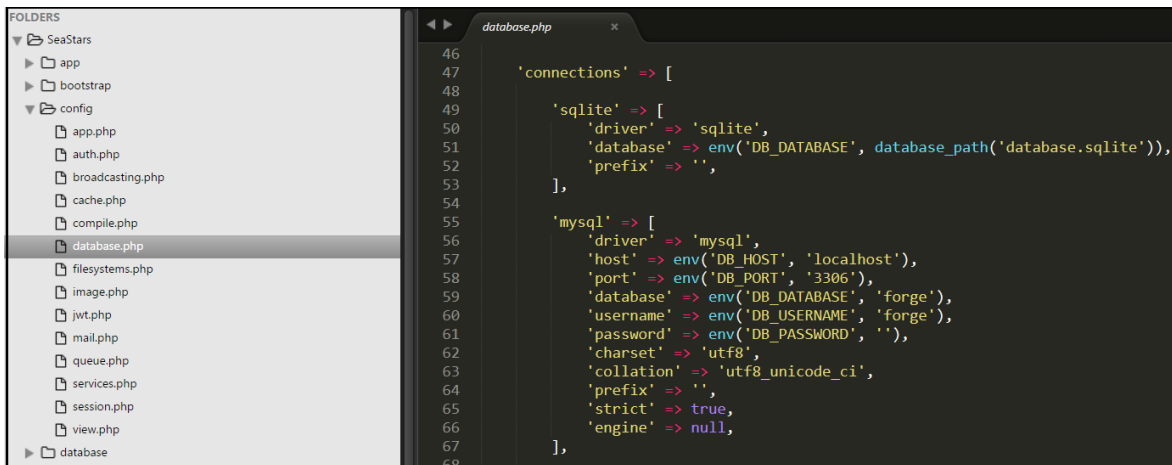


Figura 15 - Configuração do driver da BD na aplicação.

Da análise da figura conseguimos perceber que a parametrização da driver recolhe a informação que está guardada no ficheiro *.env* através do comando `env(Chave, nome_fictício)`. O Nome fictício é dado apenas por uma questão de segurança uma vez que o valor que retorna está referenciado pela chave.

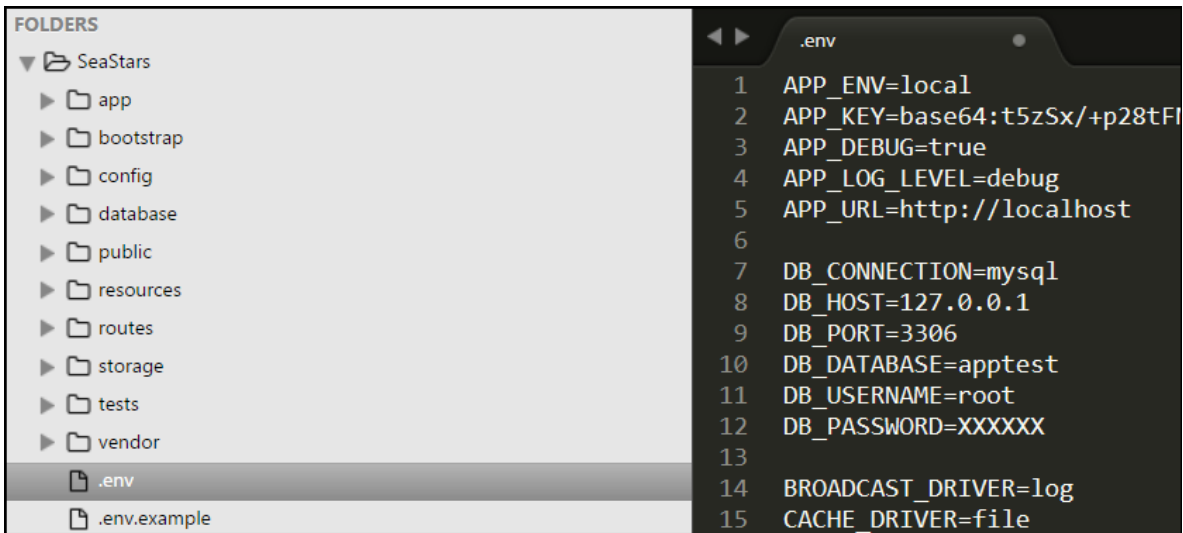


Figura 16 - Ficheiro *.env* de configuração.

Analisando a Figura 17, facilmente se percebe quais as chaves relevantes a ser chamadas na configuração do *driver*. Neste ficheiro é está definido o tipo de conexão, credenciais da base de dados, entre outros.

3.3.5. Migração do Modelo de Dados

Após o desenho de arquitectura e desenvolvimento dos modelos que implementam a lógica numa BD e respectiva configuração, é necessário migrar toda a informação relativa ao modelo de dados através do comando *php artisan migrate*. Ao correr esta linha de comando, todas as migrações do modelo de dados são executadas implementando a lógica na BD (Figuras 18 e 19).

```

C:\> Linha de comandos

D:\xampp\htdocs\SeaStars>php artisan migrate
Migrated: 2016_11_06_023335_create_users_table
Migrated: 2016_11_07_202108_create_friends_table
Migrated: 2016_11_24_234416_create_albums_table
Migrated: 2016_11_28_165841_create_pictures_table
Migrated: 2017_03_19_163119_create_markers_table
Migrated: 2017_04_13_181738_create_password_resets_table

D:\xampp\htdocs\SeaStars>
```

Figura 17 - Execução do comando para migrar o modelo de dados.

Analisando a figura anterior e a seguinte, o comando é executado e a estrutura dos modelos é implementada na BD como é possível verificar.

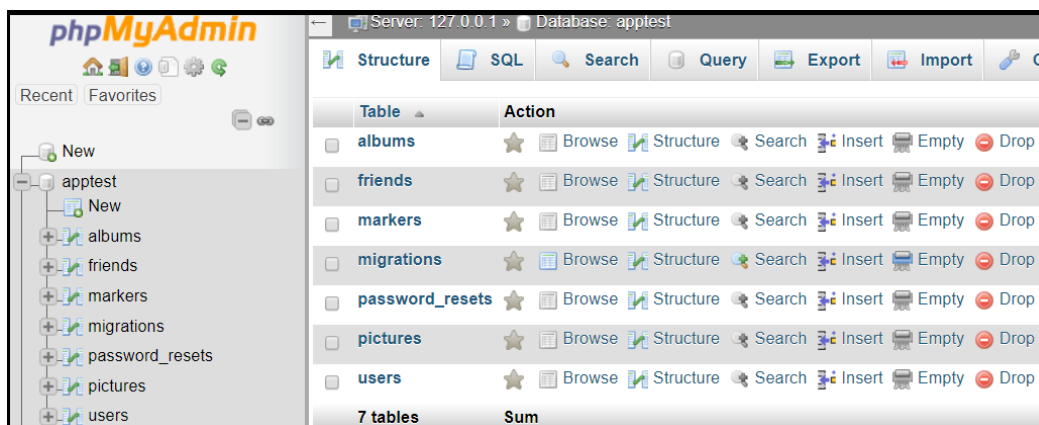


Figura 18 - Resultados depois da migração do modelo de dados.

É sempre possível voltar recuperar o que foi migrado tendo em vista futuras correcções desde que a ligação à BD não sofra alterações e seja mantida nos ficheiros de configuração. Utilizando o comando *php artisan migrate:reset* é possível migrar de volta o modelo de dados.

3.4. Funcionalidades da Aplicação Sea-Stars Watching

Nesta secção será apresentado o funcionamento da aplicação, desde a criação de uma conta, álbum de fotografias, *upload*, visualização e edição de fotografias e a interacção no mapa interativo com outros utilizadores que tenham também fotografias georreferenciadas.

De forma a evitar o uso indevido da aplicação, optou-se por um sistema de autenticação onde o utilizador começa por criar uma conta preenchendo um formulário obrigatório. Ao validar o formulário o utilizador regista-se e é autenticado de forma automática sendo redireccionado para a página inicial. Nas Figuras 20 e 21 são apresentados estes dois procedimentos.

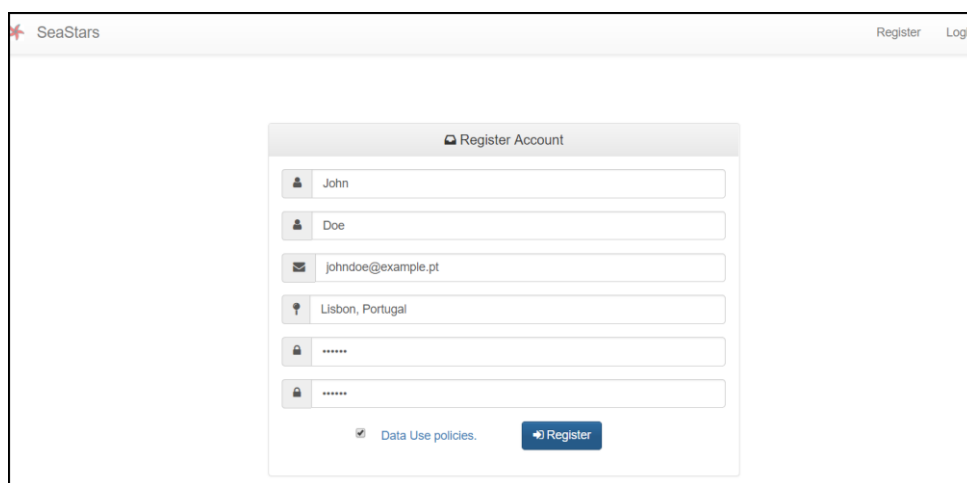
The image shows a web browser window with the SeaStars application. The main content is a 'Register Account' form. It includes input fields for a first name (John), a last name (Doe), an email address (johndoe@example.pt), a location (Lisbon, Portugal), and two password fields (both masked with asterisks). There is a checkbox for 'Data Use policies' and a blue 'Register' button. The top of the page has a 'SeaStars' logo and links for 'Register' and 'Login'.

Figura 20 - Criação de conta de utilizador.

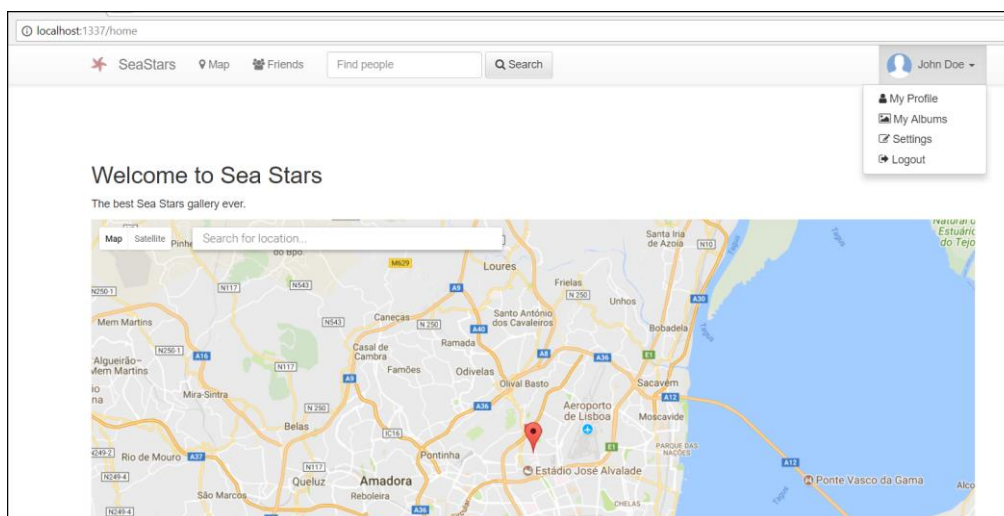


Figura 21 - Pagina inicial.

O utilizador pode criar álbuns ao seu gosto onde deverá descrever o título do álbum e a fotografia de capa. Na Figura 22 é possível visualizar este processo.

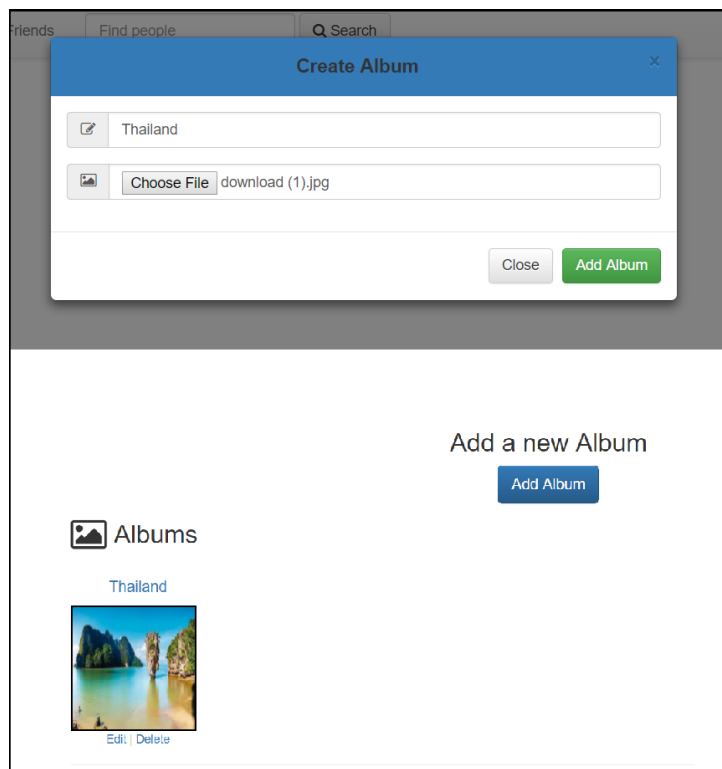


Figura 22 - Criação de um álbum novo.

Após a criação do álbum, o utilizador pode descarregar fotografias e posteriormente editá-las na sua localização e descrição. Na Figura 23 observa-se o *upload* de fotografias e a navegação pelo álbum fotográfico.

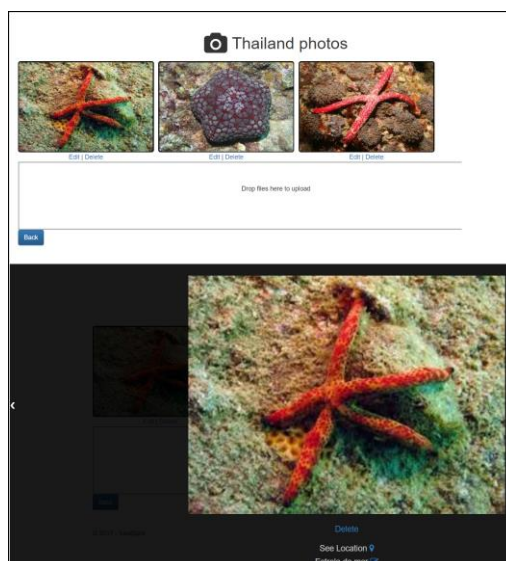


Figura 23 - Descarregamento e visualização de fotografias.

O utilizador pode editar as fotografias, actualizando a sua localização e descrição conforme local de registo e características correspondentes. Na Figura 24 observa-se a georreferenciação da fotografia onde o utilizador arrasta o ponto para um local onde foi registado o evento. Por outro lado, é feita a edição da descrição onde o utilizador descreve as características da estrela e outras componentes associadas.

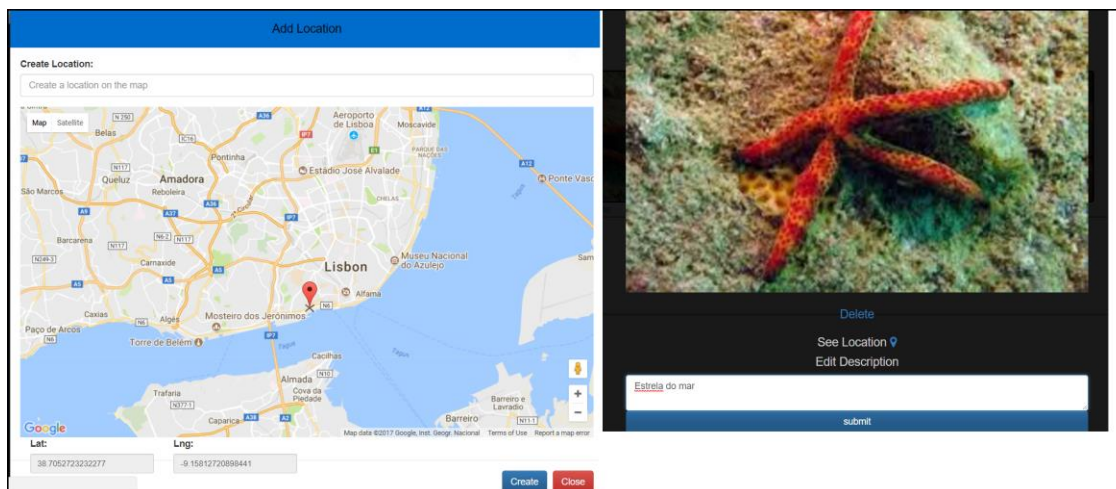


Figura 24 - Georreferenciação e edição da descrição da fotografia.

Após edição de fotografia, o utilizador pode interagir com outros utilizadores através do mapa interativo na página inicial onde poderá visualizar o perfil e álbuns associados dos outros utilizadores com conta na aplicação. Estes eventos estão associados a um símbolo que corresponde a uma estrela-do-mar que fica associado ao local onde foi registada.

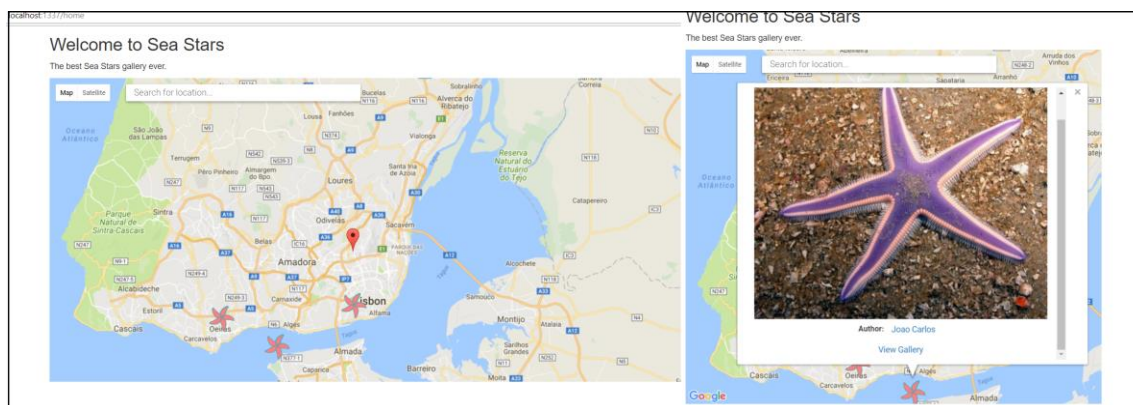


Figura 25 - Interação com o mapa interativo.

Na Figura 26 é evidenciado um exemplo de consulta de perfil e álbum de fotografias de outro utilizador através de um ponto georreferenciado. (Ver Figura 26).

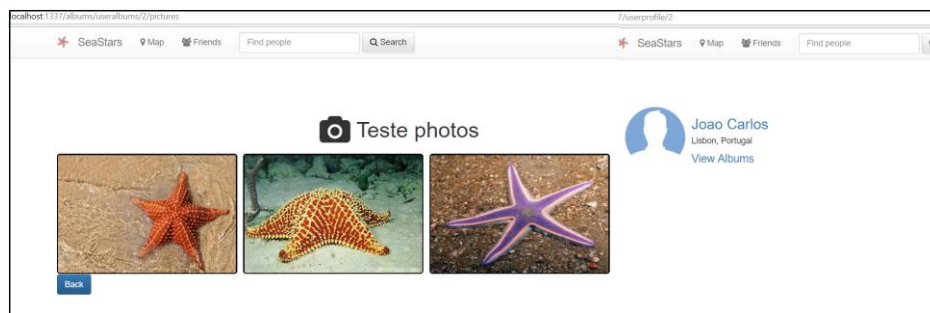


Figura 26 - Álbum e perfil de outro utilizador.

O mapa é disponibilizado graças a um serviço do *Google Maps*. A activação deste serviço requer o registo no *site* do Google para programados. Depois do registo, o Google fornece uma chave única, garantindo o acesso e activação de um vasta gama de funcionalidades disponíveis.

Além destas funcionalidades, é possível concretizar outras que foram desenvolvidas em paralelo na aplicação. Estas são por exemplo a adição de amigos que tenham conta na aplicação, alteração de perfil e pesquisa de utilizadores.

Para adicionar um utilizador como amigo, é necessário navegar até ao seu perfil onde posteriormente terá a opção de adicionar. O outro utilizador estará a espera do pedido na sua página de perfil onde pode ou não aceitar o pedido. Na Figura 27 apresenta-se a interacção com a amizade de utilizadores.

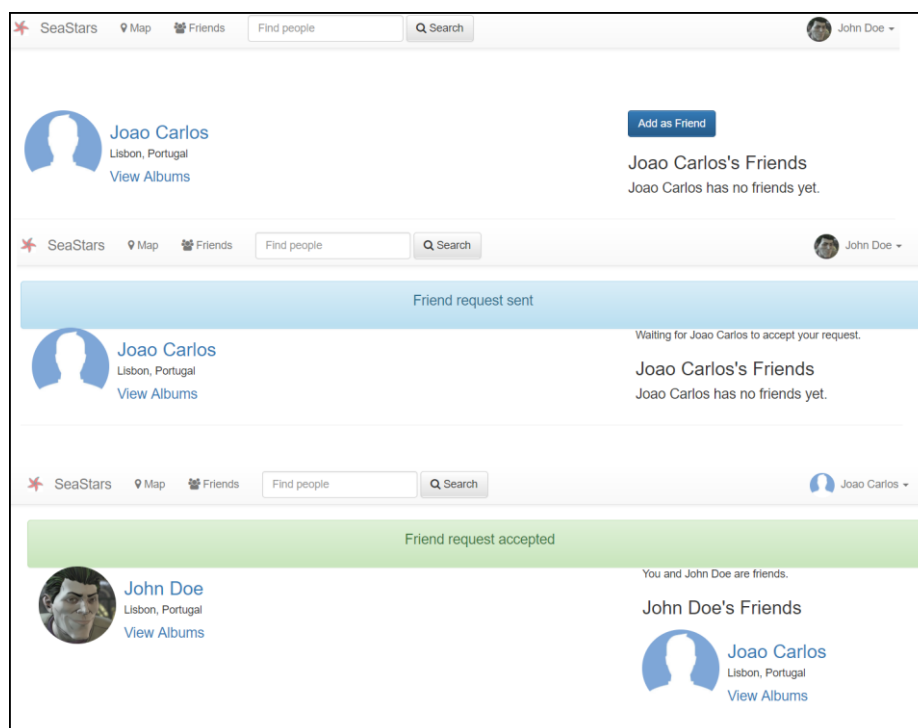


Figura 27 - Envio e aceitação de pedidos de amizade.

Para alterar o perfil navega-se até à página de configuração de perfil. Esta página contém os detalhes do utilizador autenticado, sendo possível alterar a sua fotografia e informação pessoal. Na Figura 28 é apresentado o procedimento de alteração da configuração de perfil.

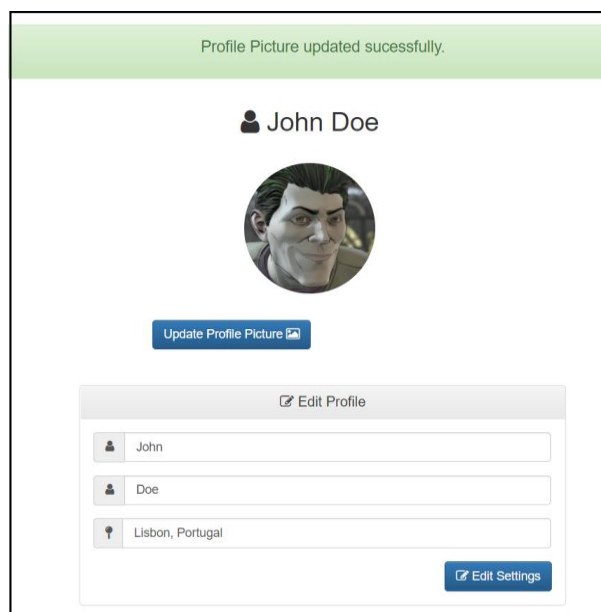


Figura 28 - Exemplo de alteração no perfil do utilizador.

Na procura por utilizadores, o utilizador autenticado pode pesquisar por nome ou letras. A resposta obtida serão todos os utilizadores que correspondem aos da pesquisa. O utilizador pode desligar a sua sessão executando o *logout* no menu de navegação. Estes dois eventos são verificados na Figura 29.

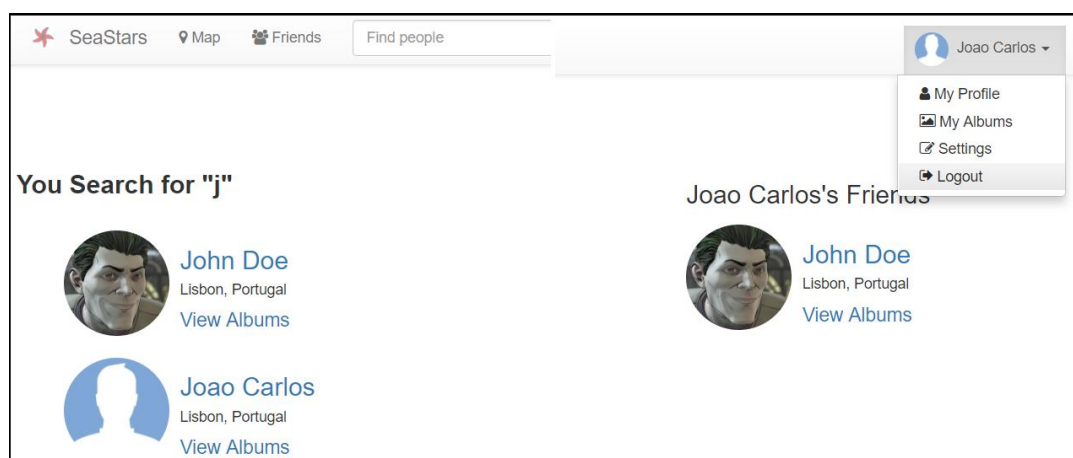


Figura 29 - Pesquisa e *logout*.

A aplicação Sea-Stars possui um estilo de formatação com padrão responsivo. Este estilo permite que a camada de apresentação ou interface gráfica se adapte consoante o tamanho do ecrã do dispositivo garantindo assim, uma aplicação otimizada para dispositivos com variados tamanhos de ecrã como *smartphones* e *tablets*. Este *design* é garantido graças a *framework* Bootstrap que contém funcionalidades a nível da linguagem CSS3 permitindo que camada de apresentação se adapte de forma automática ao tamanho de ecrã, através de classes inerentes a esta *framework* na definição dos estilos nos ficheiros HTML presentes nas Vistas (W3Schools, 2017).

Nas figuras seguintes (Figuras 30 e 31) são apresentadas as diferenças da camada de apresentação num ecrã de um computador clássico e num ecrã com tamanho mais reduzido como um *smartphone*.

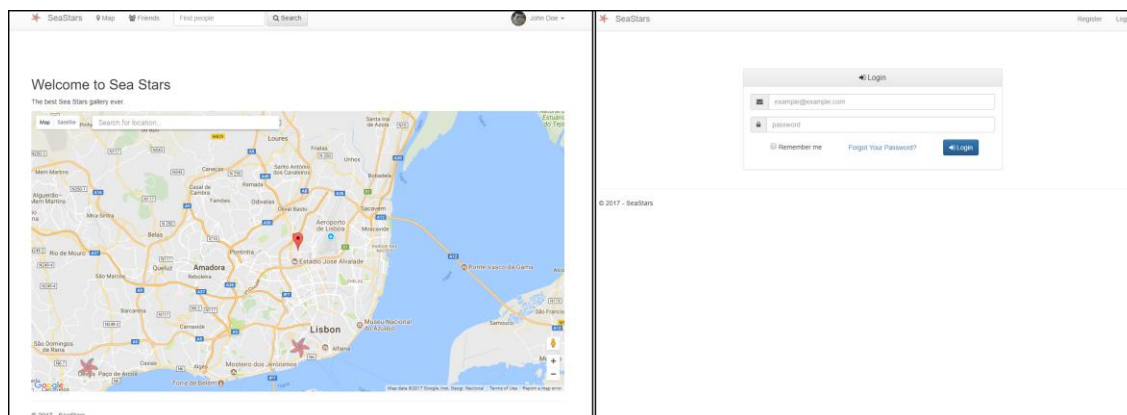


Figura 30 - Vista de um ecrã com tamanho clássico.

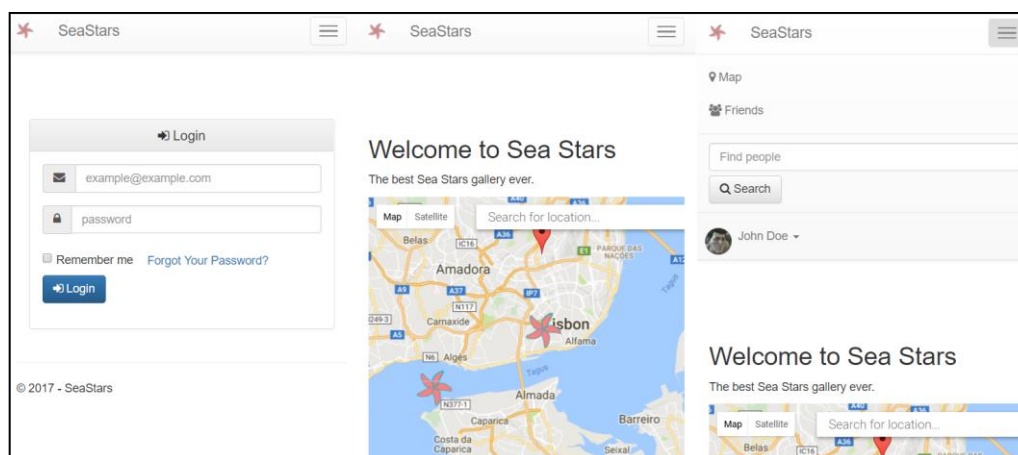


Figura 31 - Vista de um ecrã de smartphone.

3.5. Utilização dos Mapas Google

Para utilizar os serviços que permitem a disponibilização dos mapas do *Google*, é necessário criar uma conta no *site* do *Google Developers*¹⁹. Este registo foi feito utilizando uma conta de *email* da *Google*. Ao efectuar o registo de conta, é atribuída uma chave que fornece o acesso a estes serviços. Através desta chave, é disponibilizada uma vasta gama de funcionalidades como a criação do mapa na nossa aplicação, pontos georreferenciados, arrastar pontos, entre outras. Isto permitiu toda a funcionalidade existente no mapa interactivo.

3.6. Sumário

Neste capítulo foram abordados os temas da arquitectura geral, componentes lógico e técnico, estrutura REST, Sistema *Middleware*, ambiente de desenvolvimento, o funcionamento da aplicação e padrão de *design* responsivo.

Constata-se que esta arquitectura lógica é constituída por três camadas: Front-End, Back-End e a camada de interface com o utilizador em conjunto com os *web services* do *Google Maps*. A componente técnica descreve os componentes das três camadas e está subdivida pela estrutura do Modelo de dados e de software. Foi descrita a estrutura REST envolvida na aplicação bem como os recursos, métodos e resultados inerentes. O sistema *Middleware* é composto por dois grupos: Autenticados e Não Autenticados. Descreveu-se a arquitectura deste sistema onde se verificou que para cada grupo existem diferenças de autorização e quantidade de recursos disponíveis. No ambiente de desenvolvimento foram descritas as configurações da máquina virtual, instalação do projecto em ambiente Laravel, ligação da aplicação à BD, a modelação do modelo de dados e correspondentes tabelas de dados que dizem respeito às migrações. Foi explicado o funcionamento da aplicação desde a criação de uma conta de utilizador, álbum e fotografias inerentes, edição de fotografias e interacção com o mapa interactivo e outros utilizadores da aplicação. Por fim, descreveu-se o papel importante do *design* responsivo na medida em que foi apresentado a diferença de visualização para diferentes tamanhos de ecrã.

¹⁹ Google Developers <https://developers.google.com/>

Capítulo 4 – Testes e Avaliação por Perito

Estes testes permitem garantir a integridade do código, garantindo que a aplicação funciona correctamente, tendo o comportamento esperado segundo as condições de testes. Todos estes testes foram feitos localmente através da consola do *browser* e do Sublime Text onde eram verificados, validados e executados os dados de input/output com validação do responsável do projecto.

Na aplicação *Sea-Stars Watching* foram efectuados testes para garantir o correcto funcionamento dos pedidos HTTP. Cada pedido HTTP era enviado para os controladores através do Postman, ou por acesso directo ao URL das rotas com parâmetros adicionados de forma manual, ou pela submissão de formulários com interrupção de código dentro dos controladores, a fim de saber quais os dados de entrada e da resposta ao pedido. A validação final era feita em conjunto com o perito que através dos dados de resposta enviados para camada de apresentação ou para as vistas correspondiam com o que era exigido.

Nos sistemas de *Middleware*, a validação era feita através de pedidos HTTP pelo *Postman* ou por acesso directo ao URL de cada rota de forma a ver se existia um comportamento correcto de resposta. Isto é, cada pedido devia ser filtrado consoante o estado do utilizador de autenticado e não autenticado e, também, validar se cada utilizador não tinha acesso directo a rotas privadas como por exemplo na alteração de perfil de outros utilizadores.

Nas Vistas a validação foi produzida através da ferramenta *blade.php*, nas páginas onde cada utilizador tem acesso para modificar apenas o conteúdo que lhe pertence na medida em que cada utilizador só pode adicionar os seus álbuns e fotografias e não de outros utilizadores. Por exemplo, na página de criação de álbuns onde existe a possibilidade de adicionar um ou mais álbuns, a acção de adicionar é apenas visível para o utilizador que detém essa página, não permitindo aos outros utilizadores criar, editar ou apagar esse conteúdo.

Através da avaliação do perito, verificou-se que os requisitos iniciais foram cumpridos não sendo necessário a implementação de mais funcionalidades ou melhorias.

Capítulo 5 - Conclusão

Neste capítulo são descritos as principais contribuições deste projecto, as competências adquiridas, as dificuldades encontradas e possíveis melhorias para desenvolvimentos futuros.

5.1. Principais Contribuições

Nesta dissertação foi abordado o projecto *Sea-Stars Watching* com vista a desenvolver uma aplicação *Web* responsiva de forma a estar optimizada para dispositivos móveis. O objectivo é averiguar o estado de conservação das espécies de estrelas-do-mar observando a presença ou ausência de sintomas degenerativos através do registo fotográfico das estrelas criando um catálogo com os dados adquiridos e também, de igual forma, analisar o estado de conservação de outras espécies animais e vegetais. A principal contribuição neste projecto foi o desenvolvimento e instalação da aplicação *Sea-Stars Watching*.

Todos estes componentes foram implementados. A lógica do Modelo feita através dos comandos de Laravel. Os comandos do Laravel automatizam uma vasta gama de desenvolvimentos e processos que se fossem feitos de forma manual aumentavam a probabilidade de erros existentes levando à perda de maior quantidade de tempo de desenvolvimento.

De igual forma para os controladores, recorreram-se a estes comandos aumentando a rentabilidade do tempo, bastando apenas programar a lógica inerente. As Vistas foram feitas de forma manual uma vez que o Laravel não possui comandos específicos para estes componentes.

A BD foi desenhada e desenvolvida em paralelo com os modelos/entidades garantindo a consistência necessária para o sucesso das migrações das tabelas de dados para o MySQL. O sistema de rotas foi elaborado utilizando a lógica *RESTful web services* garantindo a operacionalidade deste sistema.

Mapa final apresenta pontos com uma dada latitude, longitude e conteúdo associado evidência a existência de informação geográfica no sistema de informação desenvolvido.

Os resultados dos testes em conjunto com a avaliação do perito foram positivos pois foi garantido o funcionamento correcto e exigido pelos requisitos.

5.2. Competências Adquiridas

Na realização deste projecto foi possível adquirir experiência profissional e obter novos conhecimentos tecnológicos. Este projecto permitiu angariar um vasto leque de conhecimentos em plataformas de desenvolvimento *web*, arquitecturas orientadas a serviços, desenvolvimento em linguagens Back-End e Front-End e orientadas a objectos. No decorrer deste projecto houve bastante contacto com novas tecnologias de forma a conseguir implementar as funcionalidades pretendidas e cumprir os objectivos definidos.

Na aplicação SeaStars, para a criação das Vistas e o fluxo *Model-View-Controller*, houve aprendizagem das linguagens do lado do cliente, como JavaScript/jQuery, HTML5, CSS3 e no *Web Responsive Design* através da utilização da *framework* Bootstrap. Foram efectuadas melhorias nas competências técnicas que se possuíam para desenvolvimento em PHP e criação de serviços em JSON. Na integração da estrutura REST, melhorei as minhas competências técnicas que já possuía do meu percurso académico na construção de *web services* baseados em REST.

Adquiriti experiência na criação de uma aplicação com a separação em camadas de Front-End e Back-End e Interface de Utilizador e também na arquitectura geral onde adquiri experiência na criação do Modelo de dados e no padrão de arquitectura MVC que nos dias de hoje é o mais actual e muito utilizado, tornando os desenvolvimentos muito mais organizados, metódicos e eficientes. Também foi adquirida experiência na configuração de máquinas virtuais na medida em que foi necessário configurar um servidor local para alocação da aplicação e dos dados e oque permitiu também fazer os testes de qualidade.

5.3. Dificuldades Encontradas

A maior dificuldade que senti no início foi o entendimento do modelo de dados e de como estruturar a sua arquitectura. Outra dificuldade foi identificar quais as melhores plataformas para implementar o desenvolvimento *web* e ao mesmo tempo garantir um *design* optimizado para dispositivos móveis. Estes dois pontos levaram-me a um estudo aprofundado de tecnologias e documentação relacionada.

Ao longo de tempo deparei-me com outra grande dificuldade que foi o tempo disponível. Uma vez que me encontrava a trabalhar e desenvolvia este projecto, tornou difícil a conciliação, tornando-se num obstáculo que me levou a alguns pontos de saturação e de paragens durante algum tempo no desenvolvimento tornando difícil voltar a inserir-me novamente no projecto. Outra dificuldade foi a gestão da quantidade de informação. Estando sozinho a desenvolver tornou-se difícil e moroso o desenvolvimento de algumas componentes da aplicação.

5.4. Trabalho Futuro

Na aplicação SeaStars existem alguns pontos de melhoria nomeadamente em termos de desenvolvimento *mobile* onde ficou definida a estrutura REST orientada para dispositivos móveis e o que constitui assim a API da aplicação. Foi iniciado um desenvolvimento em Java no Android Studio que não chegou a ser concluída a tempo e que seria a versão *mobile* para *smartphones* e *tablets* o que tornava mais rápida a interacção com o utilizador. Outra melhoria poderá ser o sistema *Middleware* para administradores o que poderia originar um conjunto de rotas específico e autorizações de pedidos para praticamente todas as rotas, aumentando a segurança da aplicação.

Bibliografia

Apache , 2017. *Apache Tomcat*. [Online] Disponível em: <http://tomcat.apache.org> [Acedido em Junho 2017].

Bootstrap, 2017. *Getting Started - Bootstrap*. [Online] Disponível em: <http://getbootstrap.com/getting-started> [Acedido em Maio 2017].

Composer, 2017. *Introduction - Composer*. [Online] Disponível em: <https://getcomposer.org/doc/00-intro.md> [Acedido em Maio 2017].

Couto, Francisco, 2011. *Desenvolvimento de Sistemas de Informação baseados em PHP e MySQL, Java e Oracle*. s.l.:s.n. [Acedido em Maio 2017].

CPAS, 2011. *Centro Português de Actividades Subaquáticas*. [Online] Disponível em: <http://www.cpas.pt> [Acedido em 2017].

GISGEO, 2017. *GISGEO Information Systems*. [Online] Disponível em: <http://gisgeo.pt> [Acedido em Junho 2017].

Google, 2017. *Google Maps JavaScript API / Google Developers*. [Online] Disponível em: <https://developers.google.com/maps/documentation/javascript> [Acedido em Maio 2017].

Laravel, 2017. *Configuration - Laravel - The PHP Framework For Web Artisans*. [Online] Disponível em: <https://laravel.com/docs/5.3/configuration> [Acedido em Maio 2017].

Laravel, 2017. *Installation - Laravel - The PHP Framework For Web Artisans*. [Online] Disponível em: <https://laravel.com/docs/5.3> [Acedido em Maio 2017].

Laravel, 2017. *Middleware - Laravel - The PHP Framework For Web Artisans*. [Online] Disponível em: <https://laravel.com/docs/5.3/middleware> [Acedido em Maio 2017].

Microsoft, 2008. *ASP .NET MVC Routing Overview (C#) / Microsoft Docs*. [Online] Disponível em: <https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/controllers-and-routing/asp-net-mvc-routing-overview-cs> [Acedido em Maio 2017].

Microsoft, 2017. *ASP .NET MVC Overview*. [Online] Disponível em: [https://msdn.microsoft.com/en-us/library/dd381412\(v=vs.108\).aspx](https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx) [Acedido em Maio 2017].

MySQL, 2017. *MySQL - MySQL 5.7 Reference Manual*. [Online] Disponível em: <https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html> [Acedido em Junho 2017].

Oracle, 2013. *What are RESTfull Web Services? - The Java EE 6 Tutorial*. [Online] Disponível em: <http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html> [Acedido em Maio 2017].

Oracle, 2017. *Introduction to Oracle Database*. [Online] Disponível em: https://docs.oracle.com/cd/B19306_01/server.102/b14220/intro.htm [Acedido em Junho 2017].

PhpMyAdmin, 2003-2017. *phpMyAdmin*. [Online] Disponível em: <https://www.phpmyadmin.net> [Acedido em Maio 2017].

Pluralsight, 2017. *What's the difference between the Front-end and Back-end?*. [Online] Disponível em: <https://www.pluralsight.com/blog/film-games/whats-difference-front-end-back-end> [Acedido em Junho 2017].

Postman, 2017. *Postman / Supercharge your API workflow*. [Online] Disponível em: <https://www.getpostman.com> [Acedido em Junho 2017].

ProgrammableWeb, 2014. *Review: Postman Client makes RESTfull API Exploration a breeze / Programmable Web*. [Online] Disponível em: <https://www.programmableweb.com/news/review-postman-client-makes-restful-api-exploration-breeze/brief/2014/01/27> [Acedido em Maio 2017].

SearchSQLServer, 2017. *What is Microsoft SQL Server?*. [Online] Disponível em: <http://searchsqlserver.techtarget.com/definition/SQL-Server> [Acedido em Junho 2017].

SearchWindowsServer, 2008. *What is Internet Information Services (IIS) ?*. [Online] Disponível em: <http://searchwindowsserver.techtarget.com/definition/IIS> [Acedido em Junho 2017].

Tutorials Point, 2017. *ASP .NET MVC Routing*. [Online] Disponível em: https://www.tutorialspoint.com/asp.net_mvc/asp.net_mvc_routing.htm [Acedido em Junho 2017].

Tutorials Point, 2017. *RESTful Web Services Tutorial*. [Online] Disponível em: <https://www.tutorialspoint.com/restful> [Acedido em Junho 2017].

Tutorials Point, 2017. *Web Server*. [Online] Disponível em: https://www.tutorialspoint.com/internet_technologies/web_servers.htm [Acedido em Junho 2017].

Tutorials Point, 2017. *What are Web Services?*. [Online] Disponível em: https://www.tutorialspoint.com/webservices/what_are_web_services.htm [Acedido em Junho 2017].

W3Schools, 2017. *Bootstrap 3 Tutorial*. [Online] Disponível em: <https://www.w3schools.com/bootstrap> [Acedido em Maio 2017].

XAMPP, 2017. *About XAMPP Project*. [Online] Disponível em: <https://www.apachefriends.org/about.html> [Acedido em Junho 2017].